Software-aided Service Bundling

Baida, Z.S.

2006

Software-aided Service Bundling

Intelligent Methods & Tools for Graphical Service Modeling

Ziv Baida

VRIJE UNIVERSITEIT

Software-aided Service Bundling
Intelligent Methods & Tools for Graphical Service Modeling

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. T. Sminia,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op maandag 29 mei 2006 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Ziv Steven Baida

geboren te Ramat-Gan, Israël

promotor:	prof.dr. J.M. Akkermans
copromotor:	dr. J. Gordijn

SIKS Dissertation Series No. 2006-06.
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems.

Promotiecommissie:
prof.dr. J.M. Akkermans (promotor)
dr. J. Gordijn (copromotor)
prof.dr.ir. P.M.R.J.O. Dewilde (Technische Universiteit Delft)
prof.dr. J. Mylopoulos (University of Toronto, Canada)
prof.dr. Y. Pigneur (University of Lausanne, Switzerland)
prof.dr. A.Th. Schreiber (Vrije Universiteit Amsterdam)
prof.dr. Y.-H. Tan (Vrije Universiteit Amsterdam)
dr. H. Weigand (Universiteit van Tilburg)

ISBN-10: 90-810622-1-2
ISBN-13: 978-90-810622-1-3

Cover design: Shiri Esh-Har (www.shirkan.nl)

Contents

Preface

Nowadays I feel the only constant is change;
don't you think it's strange?
While we – as nations – fight for freedom,
I think we are – as individuals – imprisoned,
or at least held captive
by our availability to instant messaging and mobile phone calls,
at any time, at any tense;
does that make sense?

As a scientist I seek to explain;
ask the question why.
Describe the logic
behind what may seem random.
Because why have a mind if not to question why?
Why have the thirst if not to drink the wine?[1]

And this is exactly what I do
in this work that lies in front of you.
My *Serviguration* ontology is a framework to explain
how to design a service bundle that serves an aim.

The title of this thesis, I wanted very much,
to be funny, as such:
"Capturing the rationale behind e-service bundling;
Who said *e* was irrational?"
But a quick survey showed
that too many people don't get the joke.
So I opted for a "mainstream" title (how sad),
and named my own company 'e-Rational' instead.

[1] Quotation from "Where is it written", Yentl

The world will be the same tomorrow.
Where I have left, others will follow.

The more I learn,
the more I realize
the less I know.[2]

Therefore all is left for me is to acknowledge
that this thesis is just another brick in the wall of knowledge.
A brick of 300 pages, of 30 months,
of ups and downs, of smiles and tears.

Now that this marathon of mine has ended,
it's time to say that I am indebted
to many who have helped, assisted and supported;
to those who pushed me towards their goals,
and to those who pushed me towards mine,
through OBELIX, through FrUX and e-Rational time.

As informal I am,
I will thank you by using your first name.
Alex, Amit, Ander, Andrei and Annette;
Carlos, Clemens, Erik, Femke and Franka;
Galit, Gil, Hanne, Hans and Hans;
Jaap, Marta, Michel, Nieves, Oren and Patricia;
Robert, Roel and Rose-Marie;
Shiri, and of course Zsofi.
Last, but always first,
great support, for every piece and every bit,
has been – and is – given by Aba, Ima, Ofra & Idith.

The word 'I' has been used here often enough;
from now on, *pluralis modestiae*, I shall be 'we'.

Ziv Baida, Amsterdam, March 2006

[2] Quotation from "A piece of sky", Yentl

Chapter 1

Introduction

Services account for a large and growing share of the total economic output in many economies. Often services are offered partially or fully via the Internet. Hence, information systems research should focus on supporting the offering and provisioning of Internet-enabled services. An important task that software should support is service bundling. A service bundle consists of more elementary services, similarly to a PC consisting of hardware components. Bundling is often required to satisfy complex customer needs, which cannot be satisfied simply by one service. Service bundling introduces extra complexity, compared to the composition of PCs and other goods. First, business logic that sets rules for composing services into service bundles is domain- and supplier specific, implicit and ill-defined. Second, services are mostly intangible: they cannot be described by physical characteristics, as is the case for PCs or mobile phones. As a result, it is hard to describe services in unambiguous terms, which has been a prerequisite for the composition of complex physical products by software. We therefore conclude that services must be described by non-physical properties, and additionally that domain- and supplier specific business rules must be modeled in order to achieve online – software-aided – service bundling.

Although services are a mature research discipline in business research, this research is mostly described in natural language, which is not suitable for software support. Research in computer science, on the other hand, provides the required formality to automate business transactions, but overlooks business logic that constitutes the essence of every service. To this end, we developed and formalized a conceptual model, an ontology, for describing services and service bundling. Content-wise, our ontology takes a business value perspective: services are economic activities. In contrast, with respect to methodology, we employ computer-based reasoning techniques to provide the necessary formality for software support of service bundling. We implemented our ontology in software tools, and provide empirical evaluation and validation through studies in complex domains: energy services and health care.

1.1 Software-aided Service Bundling

Our service ontology facilitates the automation of the service bundling task, which is traditionally performed in the minds of service personnel (employees who come in contact with customers). First and foremost one may think of automation for online service provisioning, so-called e-services. Online software applications can replace service personnel in offering customer-tailored service bundles only when the business logic behind this process is captured and formalized. As we will show in the rest of this section, online service bundling is indeed gaining importance. Second, automation can also be used offline, to increase the efficiency and effectiveness of tasks that humans perform.

1.1.1 Online Service Bundling

Online service bundling is more a future scenario than a current happening. This scenario is driven by several economic and technological forces, related separately to the *online* aspect, to the *service* aspect, and to the *bundling* aspect of *online service bundling*.

Online service bundling
The Internet is said to link together two forces in the last decennium: globalization of businesses and ICT revolution (Pohjola 2002). The Internet has changed dramatically the habits of home users as well as commercial businesses and governmental organizations. Many consider the Internet to be a key source of information, a means to lower operational costs and a channel to interact with customers. From a customer perspective as well as from a supplier perspective, the Internet has several important advantages. First, geographical boundaries that have limited the matchmaking between customers and suppliers now disappear. Second, the Internet offers flexibility in time because transactions can be done also beyond opening hours of brick-and-mortar shops and offices, offering extra ease and comfort to customers, and at the same time making it possible to generate revenues 24x7 for suppliers. Third, accessibility to many (online) shops reduces costs. Operational costs of suppliers are reduced due to a faster and machine-controlled process. From a customer perspective, shopping online and acquiring information online save costs both in terms of time (spent on searching for a product or information) and in terms of money (prices can easily be compared to find the cheapest supplier).

Online _service_ bundling
In B2C, the Internet has so far been used mainly to offer physical goods, such as books, CDs or PCs. However, from an economic perspective, services grow more and more in importance, and will be offered and deployed via the Internet increasingly. More jobs are offered by the service sector than in all other sectors of the U.S.

economy all together, and the number of jobs in the service sector is still growing. Also in middle- and low-income developing economies, the service sector accounts for the largest share of total economic output (World Trade Organization 2003).

Online service *bundling*
At the same time, globalization and a number of other economic developments and principles (that we review in Section 2.3) push companies towards a new way of marketing their goods and services: *bundling*. Bundling is "the sale of two or more separate products in one package", often for a price that is lower than the sum of the prices of the separate products in the package (Stremersch & Tellis 2002, Guiltinan 1987). The term *product* is often used to refer to goods, but in fact it is the supertype of both goods and services. 'Product' is defined as "the core output of any type of industry"; *goods* can be described as "physical objects or devices", and "*services* are actions or performances" (Lovelock 2001). The Internet enables integrating elementary services of multiple suppliers into a seamless service bundle in cases where such cooperations were not possible so far.

Online service bundling
The Internet is a popular channel mainly for selling goods. Customers can design a complex good (e.g., a PC) out of more elementary components and order such a good online, for example on the websites of Dell and Cisco. Such e-commerce scenarios are facilitated by a component-based description of goods, specifically suited for composing complex goods. So-called 'product configurators', software-based systems that compose ('design', or 'configure') complex goods out of more elementary elements - 'components' - have been on the market for several decennia (e.g., R1/XCON (McDermott 1982), MICON (Birmingham et al. 1988), VT (Gruber et al. 1996) and Cossoack (Mittal & Frayman 1989)). They share an important denominator: the components they use for composing complex artifacts are tangible, and components are configured into artifacts based on their physical properties. This is improper for services, due to their intangibility. Online design of complex services requires similar mechanisms for services.

1.1.2 Offline Service Bundling

Software support for service bundling has gained importance also for offline tasks. In the past a business would often work as an autonomous entity to offer its value proposition to its customers. Nowadays it is common practice for businesses to participate in value constellations (Porter 1985, Normann & Ramirez 1994), where several suppliers jointly offer a value proposition to customers, while they are incapable of delivering this value proposition as single businesses. This joint offering of services and goods is very much facilitated by technologies as the Internet. However, a first step in the realization of such value constellations – whereas the joint offering

is offered online or offline – is a business analysis to examine the financial feasibility of such cooperations. As typically a number of enterprises participate in a value constellation, and the business analysis has to take all participants into consideration, complexity increases. Automated reasoning by information systems can help business analysts cope with this complexity when they perform a business analysis in which service bundles are to be explored. Software tools can help business analysts design possible service bundles, and examine their financial feasibility. The choice of services to include in a service bundle determines the partners that participate in a value constellation.

1.1.3 Approach to Software-aided Service Bundling

Composing a complex artifact (e.g., service, good, software component) out of more elementary elements requires that the elementary elements are described in a way that supports composition. Service description (to facilitate composing a complex service out of more elementary services) differs from the description of physical goods (to facilitate composing a complex good) and from the description of software components (to orchestrate transactions between software components, e.g., web services, that together make up a business process). We will now discuss the differences.

Complex Services vs. Complex Goods
We take as our starting point the economic principle that when customers buy a service/good, they are interested in the value – the benefits – of that service/good, rather than in the service/good itself (Teare 1998, Lancaster 1966). Yet, if we look at websites of market leaders as Dell (PCs), Sony Ericsson (mobile phones), Amazon (books, music, movies) or Virgin (music, movies), we see that they all describe goods by their physical properties (e.g., weight, DVD zone number) or functionalities that are based on physical properties (e.g., Bluetooth wireless, video record), rather than by the benefits that they provide. Physical characteristics of goods can be expressed unambiguously, making it possible for customers and suppliers to use the same terminology when referring to a good. Hence the need to describe goods by their benefits does not arise. This is not the case with services, since services cannot be described unambiguously by their physical properties. In fact, even the same service can be different when provided twice (for example due to differences between service personnel) or can be *perceived* as different by customers due to differing customer expectations. As a result, the matchmaking between customer demands and available services cannot be performed based on physical characteristics of services. Instead, it can be performed based on the benefits that services deliver, in accordance with the above mentioned economic principle.

Complex Services vs. Complex Software Components
Neither is composing a complex service (a service bundle) the same as orchestrating

web services into an executable business process. First, the nature of the elements to compose is different. As mentioned before, we define services as economic activities where customers and suppliers exchange objects of economic value. Web services, on the other hand, are software applications that can be invoked over the Internet. Second, the level of focus is different. While service description and service bundling focus on *what* the economic nature of a business transaction is (business value perspective), web service description and web service composition/orchestration focus on *how* a business transaction is carried out (business process perspective).

Serviguration
Our approach, termed *serviguration,* considers services to be economic activities in which suppliers and customers exchange benefits, objects of economic value, such as money, valuable capabilities or experiences and possibly also physical goods. Content-wise we interpret the term service as it is interpreted in the service management and service marketing literature, which is where the term service stems from. We employ conceptual modeling and computer based reasoning techniques to facilitate software-aided reasoning on (1) the design of packages of services, and (2) the matchmaking between customers (who have certain needs) and available services and service bundles (that offer benefits to satisfy these needs). The key to answering both questions lies in considering a service as an activity of exchanging benefits. This is opposed to other approaches in computer science (Payne et al. 2001, Paolucci, Kawamura, Payne & Sycara 2002, Fox et al. 1997), where requirements (customer needs) are matched with available services based on physical properties or functionalities. Designing a service bundle is designing an artifact that provides a set of benefits, and requires the sacrifice of other benefits. Benefits are derived through an understanding of how customer demands can be satisfied by service outcomes. This results in the ability to automate the offering of customer-tailored service bundles, such that a service bundle includes services of multiple suppliers, that *together* satisfy a customer's demand, while the single services of single suppliers cannot satisfy the demand.

1.2 Research Context

We report about research conducted within the framework of two research projects: OBELIX (Ontology-Based ELectronic Integration of CompleX Products and Value Chains[1]) and FrUX (Freeband User eXperience[2]). OBELIX was an EC-funded IST project, starting March 2002 until November 2004. It focused on integration and interoperability capabilities for e-business scenarios for value constellations, characterized by complex goods, services and supply chains. FrUX started in January 2004,

[1] `http://www.cs.vu.nl/~obelix`
[2] `http://frux.freeband.nl`

and is scheduled to be completed in June 2008. As part of the Dutch national Freeband program, it is partly financed by the Dutch Ministry of Economic Affairs. The main objective of FrUX is to advance the understanding of designing and provisioning ICT-based services and service bundles that automatically adapt to changes in the context of user groups, and push information based on group profiles.

Both projects apply research directly to real-world situations. To achieve this, the projects involve research institutes as well as partners from various sectors in the industry (OBELIX involved partners from the energy sector and from the music industry; FrUX involves partners from the health sector and from the police). Consequently, our role as researchers is dual. On the one hand we perform research in an academic environment, resulting in theory formation as manifested in publications in scientific conferences and journals. On the other hand, we have a consulting role towards partners from various industries, as their interest in the project is that the research we conduct helps them achieve their business goals, or tackle business problems and challenges.

1.3 Research Question

The central research question in this thesis is:

How can services be modeled such that the task of designing service bundles can be automated?

Automation always has some goal(s). The goals of an information system determines which knowledge should be captured. In our case we are interested in automating the design of service bundles for two usages:

1. Realize websites where service bundles will be offered to customers based on their needs and demands.
2. Conduct business analyses to find financially interesting cooperations between service suppliers who can offer their independent services as a bundled offering.

From a business research perspective, our service modeling approach increases the tangibility of services, because it concretizes and formalizes the essence of the mostly intangible services and describes services in computational terms. From a computer science research perspective, our modeling approach shows that in spite of their intangibility, complex services (service bundles) can be designed – or configured – similarly to how product configurators configure complex physical goods, and taking business logic into consideration.

Given that our research takes place in a multidisciplinary environment, where topics related to business research are tackled using practices from computer- and information sciences, the above research question can be refined into several smaller research questions:

1. **What are services?** Originally, we had our own idea about how to interpret the term 'service'. Once engaged in a literature review, we found that many interpretations exist.

2. **What is the business logic behind consuming and providing service bundles?** Customers and suppliers view service bundles differently. We are interested in understanding what makes a service bundle worth consuming, from a customer perspective, and why are services combined into a service bundle, from a supplier perspective. A remark about the customer perspective is in place here. Understanding customer behavior can be the topic of numerous research projects within marketing departments in business schools, and is not the main focus of our research. Different answers may exist for this question, depending on criteria as the type of service and the type of customer. We are not interested in exploring the marketing criteria that explain a customer's preference for (an instance of, or a type of) a service, but in a more abstract understanding of the reasoning that leads customers to consume services.

3. **How can existing computer based reasoning techniques capture the above logics?** Having understood what a service is, why customers want to consume services and what is the supply-side logic behind combining services into service bundles, we are interested in examining how reasoning techniques from computer- and information science can capture the above business knowledge, such that the design of customer-centric service bundles of networked enterprises can be automated.

1.4 Research Approach

As opposed to traditional research which aims at creating knowledge only (Styhre & Sundgren 2005), our research has a dual goal: expanding scientific knowledge, as well as solving practical problems. To achieve this, we researchers engage in an interactive, collaborative process with practitioners who know a domain, to which we apply a theory. Our research approach is visualized in Figure 1.1.

Background: Business Research & Practice and Configuration Theory
Being a multidisciplinary topic, software-based service bundling from a business value perspective is too hard a nut to crack using research practices from either business research or computer science. Our approach therefore uses knowledge and

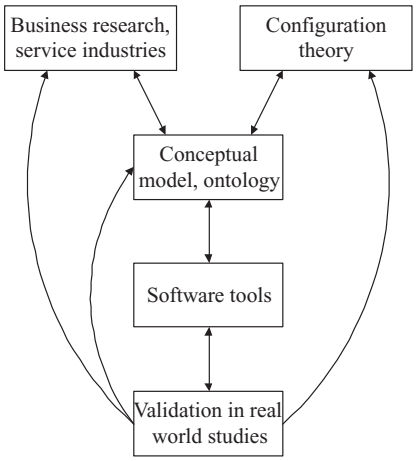

Figure 1.1: Research approach (arrows denote the direction of progress and backtracking in the research)

techniques from both disciplines in order to represent the service bundling task as a configuration task, for which computer-based solutions exist. Yet, the notion of 'configuration' is not common within business research (certainly not in the context of service bundling), and the business value perspective on services is not common within computer science. As the problem we wish to solve is of a business nature, and the notions 'service' and 'service bundling' stem from business research, this is the starting point of our research.

Conceptual Model
We draw knowledge from years of research in business schools, resulting in a wealth of literature on service management and service marketing. The vast majority of this knowledge is represented in natural language, which is unsuitable for computer reasoning. Therefore, a necessary step in solving the problem at hand is creating a conceptual model of services and service bundles, based on business research. Real-world studies from service industries serve us in developing this conceptual model. In order to represent service bundling as a configuration problem, our conceptual model adheres to and is also based on configuration theory. Conceptual models exist for configuration, as this research has been performed within computer science and AI groups, where formal conceptual modeling techniques are employed more often. The process of developing a conceptual model is cyclic, such that the model is repeatedly tested against the underlying theory and against real-world situations that we encounter in service industries.

Software Tools
We claim that the resulting conceptual model can be used for software-based service bundling. Subsequently, we formalize the model as an ontology and develop

software tools for service bundling. Tools should be able to answer possible questions of their users (in our case: offer service bundles that fit customer demands), and use the underlying conceptual model to perform the reasoning required for answering these questions. In other words, software tools serve for validating the theoretical and computational adequacy of the underlying conceptual model. If a reasoning process appears not to be possible during software development, we return to the conceptual model to investigate why. If required, the model is adapted, or the theoretical literature is studied again to find out whether the model should and can be adapted.

Validation in Real-World Studies

Finally, we cooperate with experts from service industries to investigate real-world cases by modeling them and generating service bundles using our software tools. If satisfactory service bundles are generated by the software, which is based on our conceptual model, our claim has proven to be correct. If the software generates wrong service bundles, or does not generate the desired service bundles, we re-examine the software itself, the underlying conceptual model, and if necessary also the literature on which our conceptual model is based. When necessary, we adapt the software and the underlying conceptual model. This theory formation using real-world situations through an iterative process of trying out a theory, gaining feedback and modifying the theory is required due to the ill-structured and ill-defined (in computational terms) nature of businesses, as opposed to conventional systems analysis approaches (Avison et al. 1999).

Ontology development in the context of "the ill-structured, fuzzy world of complex organizations" (Avison et al. 1999) uses real-world situations in order to develop an ontology (theory) that is applicable across a variety of such situations. The ontology has to be general enough to be applied across domains, but specific enough to enable solving business problems that practitioners and their businesses face. Every real-world situation that we investigate contributes to theory formation and validation. In each such study we researchers identify a research problem, seek for real-world situations where the problem can be expected, and then apply a theory and use a methodology to actively participate in these real-world situations. Insights gained while participating in a situation are then used to reflect on the theory, on the methodology and on the real-world situations. In our research, real-world situations need not necessarily be literally *solved* by the researcher (as suggested by the term 'therapeutic' that Baskerville & Myers (2004) use). Instead, the nature of real-world situations in which we are involved is often exploratory (e.g., business analyses), so practitioners do not expect us researchers to solve a problem with 'the best' solution, but to provide them with a means to find and explore various solutions.

1.5 Contribution

So far, research on the business value perspective of offering services and service bundles to customers has been performed by business researchers only (Barrutia Legarreta & Echebarria Miguel 2004, Berry & Parasuraman 1991, Grönroos 2000, Guiltinan 1987, Lovelock 1983, Normann 2001, Payne 1993, van Riel et al. 2001, Zeithaml & Bitner 1996). A main drawback of research done in business schools is that research results do not have a computational representation, to facilitate computer-based reasoning.

For business research, the contribution of this thesis is in providing a computational framework to model the logics behind offering services and service bundles to customers from a business value perspective, such that computer-based reasoning about this topic becomes possible. On the one hand, this thesis centers around describing knowledge that stems from business research. On the other hand, we employ techniques from information- and computer sciences to describe this knowledge. For computer science research, the contributions of this thesis are (1) in showing that product configurators, which have traditionally been used to configure tangibles (physical goods), can also be used to configure intangibles (services), and how this can be achieved; and (2) in demonstrating how business logic can be formalized and serve as the driver behind configuration tasks.

We refer to (in)tangibility with a dual meaning. From a business research perspective, services are said to be intangible, in the sense of 'not physical'. A formalized conceptual model makes services more tangible, in the sense of 'more concrete", because it describes them in computational terms. From a computer science research perspective, software has been used so far to configure tangibles (physical goods); we will show in this thesis how intangibles (services) can be modeled such that the configuration of intangibles can be performed similarly to the configuration of tangibles.

E-Service provisioning involves an intertwining of perspectives, ranging from a business value perspective to a computer science perspective. We firmly believe that successful e-service realization requires that perspectives are integrated, so that software components reflect the business nature of the transaction that they realize. Yet, research in business schools and in computer science departments is often monodisciplinary, and does not support the intertwining of perspectives. Research on the business value perspective of services and e-services is mostly described in natural language only, and therefore cannot be used in software realization. Research within computer science departments very often does not take into consideration the business nature of transactions that software realizes. In light of the above, another important contribution of this thesis is that it makes part of the business value perspective accessible for underlying perspectives by transforming ill-defined (in computational

terms) knowledge from the business value perspective into well-defined knowledge. Consequently, principles from the business value perspective can propagate to underlying perspectives in e-service realization.

The more 'tangible' contributions of our approach and research results are

1. an understanding of what services are;
2. an ontology for modeling services also from a customer perspective, while other work concentrates only on a supplier perspective on services;
3. an ontology for configuring intangibles, as opposed to traditional configuration research on the configuration of physical goods;
4. a four step method for using the ontology to perform a computer-supported business analysis for value constellations;
5. a graphical representation of the ontology to enhance communication with stakeholders; and
6. validation through implementation of software tools and through large scale real-world studies in complex domains.

1.6 Structure of this Thesis

This thesis is structured as follows:

Chapter 1 presents the domain- and research background that lead to this thesis, the research goals, the research approach and the contributions of the thesis.

Chapter 2 provides the justification for our service ontology. It investigates the use of the term 'service' in different scientific fields, the need for a formal approach to service bundling and the requirements for such a formal (ontological) approach. It shows that existing mechanisms lack characteristics that such an approach requires.

Chapter 3 presents the service ontology itself, the main output of our research.

Chapter 4 shows how service bundling can be represented as a task of configuring intangibles and tangibles, from a supplier's perspective.

Chapter 5 outlines how a customer perspective can be added to the service bundling task, so that service bundles are designed that satisfy customer demands.

Chapter 6 describes and discusses the implementation of our ontology in software tools.

Chapter 7 demonstrates the use of the service ontology through a study in the energy sector, and includes a four steps method for performing a business analysis for finding

financially interesting cooperations between service suppliers who can offer their independent services as a bundled offering.

Chapter 8 shows how our service ontology is used in the health sector to design and offer service bundles mainly for dementia patients and their (in)formal carers.

Chapter 9 summarizes the conclusions of work that has been presented in the preceding chapters, and outlines future research directions.

Appendix A includes a formal representation of inherent constraints in the service ontology. These are important for implementing information systems based on our service ontology.

1.7 Publications

Major parts of this thesis have been published in conferences and journals.

- In Baida, Akkermans & Gordijn (2003) (the Fifth International Conference on Electronic Commerce, ICEC03) we presented our research ideas, together with some early results.
- In Baida, Gordijn, Omelayenko & Akkermans (2004) (the Sixth International Conference on Electronic Commerce, ICEC04) we published a literature review on how the term 'service' is being used by different research communities.
- A discussion on service classification and how it differs from service configuration was published in Baida, Akkermans & Gordijn (2005) (workshop on Product-Related Data in Information Systems, PRODIS 2005).
- Baida, Gordijn, Morch, Sæle & Akkermans (2004) (the 17th Bled eCommerce Conference, Bled 2004) provides a four steps method for a business analysis using our service ontology, exemplified by a study in the energy sector.
- Morch, Sæle, Baida & Foss (2005) (the 8th IASTED International Conference on Power and Energy Systems, PES 2005) discusses the same study from the point of view of an energy utility.
- Akkermans, Baida, Gordijn, Peña, Altuna & Laresgoiti (2004), an article in the IEEE Intelligent Systems magazine, describes how we use our service ontology together with a configuration ontology to configure services.
- In Baida, Gordijn, Sæle, Morch & Akkermans (2004) (the 16th International Conference on Advanced Information Systems Engineering, CAiSE 2004), we

present the service offering (supply-side) perspective of our ontology, and how we use it to model energy services.

- In Baida, Gordijn, Sæle, Akkermans & Morch (2005) (the 17th International Conference on Advanced Information Systems Engineering, CAiSE 2005) we describe the service value (demand-side) perspective of our ontology, for an audience of computer scientists.

- In Baida, Gordijn, Akkermans, Sæle & Morch (2005) (the International Journal of E-Business Research) we describe the service value (demand-side) perspective of our ontology for an MIS (Management Information Systems) audience.

- Dröes, Meiland, Doruff, Varodi, Akkermans, Baida, Faber, Haaker, Moelaert, Kartseva & Tan (2005) (an article in Medical and Care Compunetics 2) is an early position paper concerning a large scale study in the health sector, where our ontology is used to develop an information system for people with dementia and their (in)formal carers.

- In Pedrinaci, Baida, Akkermans, Bernaras, Gordijn & Smithers (2005) (the 6th International Conference on Electronic Commerce and Web Technologies, EC-Web 2005) we explore ideas and show early results in one of our future research directions: a coupling between the reasoning on services from a business value perspective and the orchestration of semantic web services to carry out such services.

Part I

Context

Chapter 2

Configuring Service Bundles

***Note**: This chapter provides the justification for our service ontology. It investigates the use of the term 'service' in different scientific fields (published in the proceedings of the Sixth International Conference on Electronic Commerce ICEC04 (Baida, Gordijn, Omelayenko & Akkermans 2004)), the need for a formal approach to service bundling, and the requirements for such a formal (ontological) approach. It shows that existing mechanisms lack characteristics that such an approach requires (published in the proceedings of the Workshop on Product-Related Data in Information Systems, PRODIS 2005, (Baida, Akkermans & Gordijn 2005)).*

In this thesis we aim at software support for *service bundling*, the activity of designing a 'package' of several services that are sold as one (complex) service. To this end, a shared understanding of the term 'service' is a first pre-requisite. The multiplicity of service definitions and the variations in the use of the term 'service' (service, e-service, web service, commercial service and more) require that we first discuss this term in depth. We present a discussion of this terminology in Sections 2.1 and 2.2.

In Section 2.3 we argue why a formal approach is needed for service bundling. We show that the need for service bundling and service bundling software stems from economic and technological changes, presenting new opportunities and requirements for doing business. Software development requires that domain knowledge is described formally, so that software can reason about it. This can be achieved by an ontology with a computer-processable representation.

Consequently, in this thesis we propose a service ontology as a means to conceptualize and formalize domain knowledge on services, with the aim to automate reasoning processes. In Section 2.4 we discuss requirements for such an ontology. Sections 2.5 and 2.6 discuss the suitability of existing classification schemes for our goal. Finally, Section 2.7 provides a concluding outlook.

2.1 What is a Service?

The realization of online service offerings requires that service suppliers structure and store information and knowledge about their service offerings in a machine-readable way, so that software can reason about services. For example, software should be able to design complex services out of more elementary ones. On the one hand, information on what service offerings consist of is business knowledge, possessed by employees of service providers. It is described using concepts from the business itself (cost, value, quality level and more). On the other hand, how to model and store this information in a machine-readable way – for example to configure compositions of services – is mostly performed by information analysts. Finally, technical IT departments implement information systems that use this information.

The required involvement of three different communities in developing e-services is why different interpretations of the online service concept exist. The two extremes of business experts versus IT staff are also visible in science: business researchers and computer scientists dig into services but from an entirely different perspective. In this section we present a survey of various interpretations of 'service' to enhance mutual understanding of various disciplines involved during online service development, and communication between experts from different domains. This mutual understanding is a first step towards a comprehensive approach for online service development that reflects the multidisciplinary nature of such a development process.

2.1.1 Service Terminology

Service has become a term loaded with different meanings in different circumstances, mostly depending on who uses it. Different terms that include the word 'service', e.g., e-services, web services, commercial services etc, are referred to as just *'services'*. Also the term *'e-services'* is used with multiple interpretations.

One can classify the use of service-related terminology by various authors according to their research domain. A large community within computer scientists devotes a great deal of effort to research on a subject hardly known to business researchers: *web services*, software that can be invoked over the Internet. The same community of computer scientists often refers to *e-services* as software functionalities that are delivered via web services (Hull et al. 2003, Pires et al. 2002).

Researchers in business schools have been investigating the nature of *services* in the sense of business transactions for decades (Levitt 1973, Hill 1977, Sasser et al. 1978). They traditionally refer to 'services', without any prefix, and consider them to be economic activities, deeds and performances of a mostly intangible nature (Grönroos 2000, Kotler 1988, Zeithaml et al. 1990, Kasper et al. 1999). The term 'activity'

should be interpreted as an act of economic nature, rather than an operational process. In recent years the term *e-services* has gained ground also in the business research community (Rust & Kannan 2003, Stafford 2003, Taylor & Hunter 2002), but with a different meaning than the same term has among computer scientists. The difference in interpretations was well expressed by Stafford (2003): "Marketers see e-services as a natural outgrowth of e-commerce, but they also view services through a product-oriented lens; this is only natural. Technologists naturally view e-services as Web-delivered software functionality, often characterized under the rubric of 'web services'."

Information science researchers are trapped between these two worlds. In an attempt to bridge the gap between computer scientists and business researchers, the use of any of the above-mentioned terms is likely to fail either in the computer science community or in the business research community. Publications of information scientists often refer to 'services' (Ardissono et al. 2002, O'Sullivan et al. 2002*a*), like in the business research community (thereby possibly creating misunderstanding among readers from the web services community). Others use the term *real-world services* to differentiate it from web services (Baida, Akkermans & Gordijn 2003, Akkermans et al. 2004), or the term *commercial services* (O'Sullivan et al. 2002*b*).

All these terms – to which we refer as 'service terms' – relate to the essence of a service. Other terms are common as well, e.g., IT services, information services, public services, governmental services, and more. These consider the domain-related *contents* of the service, rather than the *definition* of what a service is. Consequently, they are not part of our discussion.

2.1.2 Services in Business Research

Services have traditionally been a topic of research among business researchers[1]. As services are now being offered *electronically* over the Internet, in recent years *e-service* research has been emerging; researchers use traditional service research as a basis for this new paradigm, and investigate differences between the "old world" and the "new" one.

Although various researchers (naturally) use different definitions for the term 'service', the service area in business research shows a consensus on many points. Representative definitions of what a service is often contain the same recurring elements. For example:

- Kotler (1988): "... any act or performance that one party can offer to another that is essentially intangible ...".

[1] When referring to 'business research' in the context of services, we consider mainly the service management and service marketing community.

- Zeithaml & Bitner (1996): "... services are deeds, processes and performances ...".
- Grönroos (2000): "... activities ... of a more or less intangible nature that normally ... take place in the interaction between the customer and service employees and/or physical resources or goods and/or systems of the service provider, which are provided as solutions to customer problems".
- Lovelock (2001): "... economic activities ... bringing about a desired change ...".

Edvardsson et al. (2005) performed an extensive literature research on service definition and service characteristics in service research literature, and incorporated in their study also input from 11 leading scholars from the service research field. They conclude that "definitions focus on the offering to the customers" and stress value-in-use. In the business research community it is thus accepted that 'services' are economic activities that often result in intangible outcomes or benefits (customer value); they are offered by a service provider to its (potential) customers. By using the term 'economic activity' (to which we also refer as 'business activity') we emphasize a business value perspective on services (regarding a service as a transaction in which customers and supplier(s) exchange objects of economic value), rather than an operational perspective (regarding a service as processes in which humans and machines produce a product through some process). We refer to this interpretation when we use the term 'service' (with no prefix) in this thesis.

Another remark on terminology is in place here. As said, services have an intangible nature. Their intangibility differentiates them from *goods*. Whereas people often consider 'good' to be a synonym of 'product', in business research literature 'product' is defined as "the core output of any type of industry", and "*goods* can be described as physical objects or devices, whereas *services* are actions or performances" (Lovelock 2001). Both 'good' and 'service' are thus subtypes of products. Hence, in this thesis we refer to goods as well as services when we use the term *products*. It should be noted that governmental and non-profit services are not excluded from our definition of services as economic activities. Also these services involve the exchange of value objects, but unlike most commercial services, in these cases the value may be ideological, political or social.

As various industries – e.g., manufacturing industries, service industries and governments – have been moving towards a broad use of the Internet instead of traditionally 'physical' processes, the business research community has adopted a new field of research: *e-services*. We identify three views on e-service definition within the business research community. First, several e-service researchers base their understanding of what e-services are on Zeithaml et al. (2000). They consider e-services to be services (interpreted as presented earlier in this section), where the Internet is used as a

channel to interact with customers (Janda et al. 2002, van Riel et al. 2001). Second, de Ruyter et al. (2001) compared several conceptualizations of e-services, and concluded that a recurring theme in these conceptualizations is integration, the seamless incorporation of technology and customer-oriented functions within the firm. They define e-services as "an interactive, content-centered and Internet-based customer service, driven by the customer and integrated with related organizational customer support processes and technologies with the goal of strengthening the customer-service provider relationship." Finally, Rust & Kannan (2003) define e-services as "the provisioning of services over electronic networks", whereby 'electronic networks' include not only the Internet, but also wireless networks as well as electronic environments such as ATMs and smart card networks, kiosks, and "all touch points with customers". This definition is centered around the statement that this emerging paradigm – e-service – is based on expanding revenues through enhancing service and building profitable customer relationships. To summarize these definitions, we can say that the first and the second definitions agree on e-services being an Internet-based version of traditional services. The first definition is not as broad as the second one in the sense that it does not mention customer relationships or business processes. The third definition includes the second one, and does not limit itself to the web.

The term *web service* is not often used in business research literature. If used by business researchers, this term is either acknowledged as a computer science term (and the computer science definition, given in the next section, is adopted (Stafford 2003)), or it is understood as services (in their business definition) delivered via the web.

To conclude our discussion on the business research community, we can say that:

- There is a broad consensus on ('traditional') *service* definition.
- Most researchers define *e-services* as an Internet-based version of 'traditional' services. Broader, and other definitions exist as well.
- *Web services* are not often referred to; when this term is being used, the computer science definition is adopted.

2.1.3 Services in Computer Science

Three 'service terms' are common in computer science[2]: *web services, e-services* and *services*.

Web services are a hot item among computer scientists. Publications of web services and semantic web researchers discuss every possible aspect of web services.

[2]We refer mainly to the semantic web community, where services are a main topic of research.

Nevertheless, research of the Gartner Group identified a widespread misunderstanding of what web services are (Hotle 2003), leading to the assumption that people mistake web services for software that is accessed over the Web, rather than software that *"implements coarsely-grained business functions*, and is accessible over the Internet", or "commonly used business processes delivered over the Web". Ample definitions of web services exist (RosettaNet consortium 2003, IBM 2000, Stencil Group 2001, Pires et al. 2002, Austin et al. 2004, Tidwell 2000). Some are implementation-oriented (e.g., the W3C defines a web service as "a software system identified by a URI[3], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols" (Austin et al. 2004)). Others use a higher level of abstraction (e.g., the Stencil Group defines them as "loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols" (Stencil Group 2001)).

Three elements are common to many web service definitions: (1) software / applications, (2) functionalities and (3) the Internet/Internet technologies. Table 2.1 summarizes the recurring elements that appear in representative web services definitions. All of the definitions agree on the fact that web services are software / applications to be used on the Internet. Most of them explicitly recognize the existence of functionalities behind the software, but not the existence of business processes or *business* functionalities. Nevertheless, the software is an implementation of generic functionalities, often offered by businesses to other businesses, to carry out some business process, that realizes a business (economic) activity. These functionalities can roughly be categorized as *information-providing services*, such as flight information providers, temperature sensors, and cameras, and *world-altering services*, such as flight-booking programs, sensor controllers, and a variety of e-commerce and business-to-business applications (McIlraith et al. 2001).

The term *e-services* has a somewhat stronger business flavor than its counterpart *web services*. E-Service definitions are characterized by a lower degree of consensus among those who use them. Govindarajan et al. (2001) write that "web services, or e-services are…", implying that web services – which were defined as software/applications – and e-services are synonyms. On the other hand, Kotov (2001) describes e-services as "the realization of federated and dynamic e-business components in the Internet environment", not putting the emphasis on *how* these e-business components are realized (i.e., by applications).

Both web services and e-services are often referred to as simply *services*. Many authors first use the terms web services or e-services, and further refer to them as

[3] `http://www.ietf.org/rfc/rfc2396.txt`

Table 2.1: Recurring elements in web service definitions

	W3C	*Stencil Group*	*RosettaNet*	*IBM*
Software / application	×	×	×	×
Functionalities		×	×	×
Business processes			×	×
Internet	×	×	×	×
XML as a supporting technology	×	×	×	×

'services', or include the term 'service' without any prefix in the title of their papers (Doulkeridis et al. 2003, Salzmann & Schätz 2003). Others adopt a business-flavored service definition, considering services as intangible goods (Chan et al. 2001). Telecommunication publications often discuss 'services' as well, either in their business interpretation (what does a supplier offer to customers, see Section 2.1.2) or as network sessions that realize these service offerings (van Halteren et al. 1999, Koutsopoulou et al. 2001).

Business services (Glushko et al. 1999, Johannesson et al. 2000) is another 'service term' that researchers in the computer science community use, although to a much lesser degree than *web services, e-services* and *services*. Since it is neither used often, nor defined, our literature review yielded no conclusions on how it is interpreted. One could assume that authors who use this term adopt a business definition for 'services', as presented in the previous section.

To conclude the discussion on computer science, we can say that:

- The term *web service* appears to be well-defined within computer science (see Stencil Group (2001)).
- *E-Service* definition is not characterized by a consensus.
- The term *service* is used as a synonym for 'web service', as well as 'e-service'. Some, on the other hand, give it a business definition: intangible products. Once again, misunderstandings are likely to happen.
- The term *business service* is sometimes used, though not defined.

It appears thus that the term *e-service* is not well-defined in either business research or computer science. Other terms, on the other hand, are well-defined within one of these communities: *services* within business research (a business activity, emphasizing its intangibility and its business value) and *web services* within computer science (emphasizing software and technologies).

2.1.4 Services in Information Science

As research on web services is becoming more and more popular among computer scientists, and service research in business schools enters maturity, researchers from information sciences try to convey a message to both communities, where the same terms have differing meanings, as we have seen. Researchers from this field often use definitions given by both other communities: when referring to software and to technologies, they use the term *web services*, as done by computer scientists, and when referring to economic activities with mostly intangible results, they use the term *services*, as their colleagues from business schools do. Representative definitions for 'service' are given in Dumas et al. (2001): "a simple or a complex task or activity, executed within an organisation on behalf of a customer or organisation", and in O'Sullivan et al. (2002*a*): "an action performed by one entity on behalf of another. This action involves the transfer of value". Sometimes authors refer to 'products and services' (Ardissono et al. 2002, Mohan & Ramesh 2003). By doing so, it becomes clear that they refer to 'services' in their business interpretation, since the comparison of services vs. products (actually meaning *goods*, rather than *products*) stems from business research. In other cases (Edmond & ter Hofstede 2000), the term 'service' is used with no definition.

In an attempt to avoid misunderstandings as a result of the term 'service' being interpreted as 'web service' rather than as an economic activity, Baida, Akkermans & Gordijn (2003) introduced the term *real-world service*, giving it the same meaning as the term 'service' has in the business research community. Others sometimes use the term *commercial service* instead of 'service' (Lenk 1995), although the discussion would be valid in case of governmental services too. Neither 'real-world service' nor 'commercial service' are used often.

Since the term *e-service* is not as mature as 'service' or even 'web service', it comes as no surprise that researchers from information sciences interpret e-services in different ways. Some define it in a way similar to the business world: "services that are delivered electronically, typically through the Internet" (Mohan & Ramesh 2003). Others consider e-service to be a synonym of web service, as often the case in computer science: "electronic services offered over the Internet are also referred to as electronic services, web services, Internet services, web-based services or e-services" (Tut & Edmond 2002).

We can conclude the discussion on information sciences by stating that:

- The term *web service* is used like in computer science.
- The term *service* is mostly used like in business research.
- *E-Services* are interpreted either as an Internet-based version of 'traditional' services (similar to many researchers from business schools), or as web services (similar to many computer scientists).
- The terms *commercial service* and *real-world service* are sometimes used to refer to services in their business interpretation.

2.2 Service Terminology: Summary

Table 2.2 summarizes the discussion on *service, e-service* and *web service*, the three most widely used 'service terms'. As the table shows, a shared understanding exists of what 'web services' stand for. Misunderstandings are likely to occur when (1) using the term 'services' within computer science or possibly information science; (2) using the term 'e-services' in any research community; and (3) discussing service-related subjects with experts from different communities (computer scientists, business researchers, information scientists). Our survey adds to existing research in providing researchers from three communities an overview of different interpretations for the terminology they use, as well as how different terms are related.

Services
Since our research applies a business value perspective to services, as done in business research, we adopt the definition of 'service' as it is understood in the business research community. Services typically present a bundle of benefits (e.g., rights, experiences, goods), for which customers are willing to sacrifice (in money, in a commitment to consume a service from a certain supplier for a long period). The benefits and the sacrifices are objects of economic value, exchanged by customers and suppliers in economic activities: services.

E-Services
We consider e-services to be services (interpreted as presented above), where the Internet is used as a channel to interact with customers in at least part of the value chain, including marketing, sales, logistics, production, delivery (in the case of services, production and consumption are inseparable) and after-sales support. When these business activities are performed online, supported by technologies as web services, we refer to them as e-services. We consider 'e-services' to be a subset of 'services'.

Table 2.2: Service terms usage: summary

	Services	*E-Services*	*Web services*
Business research	Well-defined	Core interpretation is shared; interpretations vary in the extent of generalization	Rarely used, definition borrowed from computer science
Computer science	Divergent interpretations	Technical or business definition	Well-defined
Information science	Mostly business definition	Business or technical definition	Well-defined

Web-services
As this term stems from computer science, we adopt an interpretation of web services from that discipline. Different definitions exist for web services; the definition of the Stencil Group is representative: web services are "loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols" (Stencil Group 2001). Web services are a technological means to realize e-services.

2.3 The Need for a Formal Approach to Service Bundling

We mentioned earlier that when several services are sold together as a 'package', we refer to this package as a 'service bundle'. A service bundle may include multiple services of one supplier, or services of different suppliers. Service bundling is the activity of composing various services into a service bundle. In this section we argue for a formal description of the service domain, to be used for software-aided service bundling. In particular, we will show that the need for a formal approach exists for at least two different goals: (1) the realization of complex online service offerings (e-services), and (2) business analyses of networked-enterprises. In both cases the emphasis is put on scenarios in which a set of independent services is offered by a group of suppliers as one package, a *service bundle*. In Section 2.3.2 we argue that a need for bundling exists. In Sections 2.3.3 and 2.3.4 we argue for supporting service bundling by a formal approach, and in Section 2.3.5 we discuss the desired degree of formality.

2.3.1 What is a Service Bundle?

The idea of bundling services and goods into a single 'package' is a conventional one in the business research literature, and widely used in many industries. Yet, there is no consensus on how to define the term *bundling* (Stremersch & Tellis 2002). In spite of the lack of a common definition, most definitions share a common core, as defined by Stremersch & Tellis (2002): "bundling is the sale of two or more separate products in one package" (in their discussion on pricing and bundling strategies, Stremersch & Tellis (2002) make a distinction between product bundling and price bundling, but that is beyond the scope of the current discussion). Another well-accepted definition is given by Guiltinan (1987), who refers also to the price of a bundle in his definition: bundling is "the practice of marketing two or more products or services in a single package for a special price".

Researchers from business schools distinguish between two cases: offering services only as a bundle (*pure bundling*), and offering both a bundle and its separate elements (*mixed bundling*) (Guiltinan 1987, Normann 2001, Barrutia Legarreta & Echebarria Miguel 2004).

Bundling does not necessarily refer to services. A well-known example of bundling physical goods is the selling of a PC that a customer can design by combining separate elements, including a processor, a motherboard, internal memory, a CD/DVD drive, and even a printer. Lovelock (2001) points out that many services are sold with physical goods without being charged separately, and that many services are in fact a bundle of more elementary services, possibly with physical goods as well.

In this thesis we define a *service bundle* as a package of one (the trivial case) or more services, whereby: (1) 'service' is interpreted as an economic activity in which customers and suppliers exchange objects of economic value; and (2) numerous service providers can supply the different services included in a bundle; and (3) the package is marketed and sold to customers as one whole (the elements in a service bundle are often also marketed and sold independently). An *e-service bundle* is a service bundle, where the bundle's elements as well as the bundle itself are e-services (as defined in Section 2.1.1). *Service bundling* is the marketing and sale of two or more separate services in one package (Stremersch & Tellis 2002). Finally, *e-service bundling* is the marketing and sale of two or more separate e-services in one package. The terms *service bundling* and *e-service bundling* can also be interpreted as the process of designing service bundles and e-service bundles.

2.3.2 Reasons to Bundle Services

Businesses deploy the principle of bundling for divergent reasons, as has been studied by business researchers. Guiltinan (1987) explains that from a managerial perspec-

tive, bundling is traditionally related to two phenomena:

1. Most service suppliers have a high degree of cost sharing, so that the marginal costs of selling extra services to the same customer are low. Bundling multiple services is then a means to reduce costs. The various activities (services) that are packaged into a bundle may use common processes, technological and human infrastructure of service suppliers. Once combined into a bundle, the cost of providing several services to one customer is lower than the cost of providing the same services separately to different customers.

2. Services are often interdependent in demand, meaning that a customer is often interested in more than one service. This is mostly the case for related services (e.g., bundling a set of financial services).

In more recent work, Mourdoukoutas & Mourdoukoutas (2004) argue that the semi-global economy requires companies to bundle services. In contradiction with earlier thoughts about globalization, the world economy cannot turn into a single integrated market, with the same products for domestic and international consumption (Mourdoukoutas & Mourdoukoutas 2004). Instead, globalization is combined with localization, to achieve the best of both, resulting in what is referred to as semi-global economy. On the one hand, achieving economies of scale and reducing costs requires that global businesses cooperate with their global competitors, as can be seen in the cooperation between Daimler Chrysler, Mitsubishi and Hyundai (Mourdoukoutas & Mourdoukoutas 2004). On the other hand, achieving local product differentiation requires that these global businesses compete with the same global partners by cooperating with local service providers that can help them differentiate their products, and make these products fit the local demands. Opting for this strategy is very beneficial when bundling commodity products – that are hard to differentiate – with local services. Commodities, highly standardized products as electricity, chemicals and by now also online banking services, are hard to market, since the differences between the products of competing suppliers are minimal, and often also the prices are almost the same. In such situations suppliers may want to bundle these standardized products with other products that help them differentiate themselves and compete in the market. In fact, due to the existence of an extra – non-standardized – product, customers are willing to consume the standardized product as well.

Increasing revenues has traditionally been acknowledged as a main reason for product bundling (Zhu & MacQuarrie 2003), as can be illustrated by the example of the Microsoft Office Basic suite, where three programs (Word, Excel and Outlook) are sold in the Netherlands for a single price of 210 Euro. Customer A perceives Word to be worth 150 Euro, but is not willing to pay more than 60 Euro for Excel, and is not at all interested in Outlook. Customer B perceives each of the three programs to be worth 70 Euro. If priced separately, Microsoft maximizes revenue with a price of

150 Euro for Word, 60 Euro for Excel and 70 Euro for Outlook, with total revenue equaling 340 Euro (Customer A will buy Word and Excel; customer B will buy only Excel and Outlook). But if Word, Excel and Outlook are bundled together and priced at 210 Euro, both customers will purchase the bundle, and revenue will equal 420 Euro.

Bennett & Robson (2001) discuss bundling as a means to increase competitiveness through creating entry barriers. Potential new entrants to an industry (companies that will offer the same, similar or substitute products) are one of the influencing factors of a firm's competitiveness, as shown in Michael Porter's Five Forces model for performing analyses of competition within industries (Porter 1980). A firm can increase its competitiveness by creating obstacles for potential new entrants. These obstacles are referred to as *entry barriers*. Porter (1980) discusses several ways to create entry barriers, including economies of scale, product differentiation and access to distribution channels. These are entry barriers, because they are strategic advantages that potential new entrants need to achieve in order to compete with existing actors in the industry, who have already acquired these advantages. For example, if a new entrant offers the same products as existing suppliers, but does not succeed in achieving the same economies of scale as existing suppliers do, it will not be competitive. Bennett & Robson (2001) argue that bundling also creates entry barriers: a potential new entrant will have to offer a whole set of services – and not just a single one – in order to be competitive in a given industry. Hence, by offering a bundle of services firms make it more difficult for potential new entrants, and thereby increase their own competitiveness.

Normann (2001) discusses another reason for bundling: a development towards a need-oriented matching between activities of customers and suppliers. This phenomenon stems from the deregulation of markets, as experienced by a broad variety of markets (e.g., the US airline industry was deregulated in 1978; the European electricity industry was deregulated in 1999). Owing to deregulation, the relations between customers and suppliers are no longer monopoly-based and supplier-dominated. Instead, companies compete on customers, and are required to adapt their offerings to suit the demands of customers. This, in turn, has caused two different trends (Normann 2001):

- *Unbundling*: in situations where a monopolist could force customers to buy a bundle, the elements of such a bundle are now sold also independently, because new competitors offer them independently.
- *(Re)bundling*: recombining the separate service elements into bundles that best fit customers demands. As the interaction between customers and suppliers becomes more customer need-oriented (and not – as done before regulation – monopolist need-oriented), suppliers bundle services to create an optimal solution for a customer's need.

In short, due to a number of economic developments it is beneficial – or even required – for businesses to bundle services (and goods). Yet, a bundle needs to be presented also to customers as more beneficial for them than the separate elements. To this end, suppliers can choose to emphasize the complete solution that a bundle gives to a customer need, or the price of a bundle. Accordingly, the price of a bundle is typically lower than the sum of the prices of the separate elements.

It is important to acknowledge that in the above discussion services are interpreted as economic activities that offer customers a solution for their needs, in accordance with service definition in business research and in this thesis (see Section 2.1.2). Service bundles are therefore a bundle of economic activities, offering a bundle of benefits (Kasper et al. 1999). Inherent to understanding services as economic activities that provide benefits, they are seen as activities in which customers and suppliers exchange objects of economic value, such that both perceive the value that they receive as greater than the value they give. Consequently, bundling services must also be seen as the bundling of activities that provide values and require other values. This observation is important for the positioning of the rest of our work, because it implies that a service bundle is considered as a complex activity of exchanging economic values between customers and suppliers, and not as a complex business process, focusing on operations. Bundling refers to combining economic activities, and not to combining business processes.

Economic activities are carried out by business processes. The former describe a value exchange between customer and supplier, and the latter describe how the value exchange is executed in terms of scheduling, resource flow (information, human resources) and customer-supplier interaction. Often bundled economic activities use shared business processes, so the operationalization of a service bundle requires less resources than the operationalization of the separate elements in a bundle. In this thesis we consider only the bundling of services as economic activities, and we do not discuss how a complex business process can be designed to carry out a service bundle.

2.3.3 Realizing E-Service Offerings

More and more businesses nowadays offer their services via the Internet, either parallel to or instead of the traditional physical channels. Statistics show an immense growth in the percentage of households with Internet access that actually shop online; from 27% in 1998 to nearly 50% in 2000 (Xue et al. 2003). Almost 30% of Internet users in the EU use online banking services, with the Nordic countries as leaders; nearly 65% of Internet users in Finland use online banking (Centeno 2003). Airlines sell more and more tickets online instead of through traditional travel agencies; check-in is performed online rather than at the check-in counter in the airport.

Companies as DHL and FedEx allow customers to follow their shipments through a so-called "track and trace" system. Governments start considering online voting. These are all examples showing the dominant and growing role and importance of e-services in a variety of industries.

Online service offerings introduce several new challenges, with which traditional, brick-and-mortar service providers do not have to deal.

1. **A shared understanding** of the term 'service' is required in order to enable customers search and compare services of multiple suppliers. This holds for B2C scenarios, but also for B2B scenarios where businesses form a supply chain, deploying Internet as a means to achieve efficiency gains in supply chain management, thereby relying on information integration along this chain. To arrive at such an integration, each party in the chain should have the same understanding of the good or service to be delivered.

2. **Demand-side (customer) perspective.** In e-service offerings it is no longer sufficient that only service personnel understands customers' needs; if a supplier wishes to offer customized services through an automated online process, software must be able to reason about these customer needs and about the possible service offerings satisfying such needs, so that the whole process can be provided online.

3. **Supply-side (supplier) perspective.** Online service provisioning implies that also supply-side business logic needs to be dealt with by software, to design a service offering (a set of one or more services) that adheres to business rules. These may include legislative restrictions, strategic business decisions as the choice of preferred business partners and business efficiency: the ability to use well existing infrastructures and other resources. While these considerations are all taken into account by service- and marketing personnel, as soon as the front office "goes online", software has to use this knowledge in the process of defining customer-tailored service offerings.

4. **Bundling.** The need for a software-based process becomes even greater when multiple services *and* multiple suppliers are involved in a single scenario. Consider a customer who wants to buy a service bundle of which the separate services are offered by different suppliers. Each supplier offers its added value, and together suppliers provide a complete answer for a customer need. In such a case, software should be able to decide whether and how to combine services of multiple suppliers into one service bundle. In other words, supply-side business logic that is used to create such service bundles also needs to be made machine interpretable, so that software can generate service bundles for a given customer.

In conclusion, complex online service provisioning scenarios (involving a variety of services, supplied by a variety of suppliers) require a shared understanding of the service concept, and a formalization of domain knowledge, including (1) customer needs, (2) an understanding of how available services can satisfy customer needs, and (3) the supply-side business logic that is used in making the decision to offer certain – single or bundled – services to a customer.

We suggest a formal description of services from a supplier perspective and from a customer perspective, including the business logic that dictates how to design a service offering for a given customer, or customer type. Such a formal description shall be expressed in machine-interpretable standards, as a means to express a shared understanding for the sake of automation. It is important that this service description represents the logic of both a customer perspective and a supplier perspective, because every service must present benefits to both sides, and will be consumed only if it is based on business logic of both these stakeholders. While the supplier perspective is often dealt with, in computer science the customer perspective is often not integrated in software realization.

2.3.4 Business Analysis

In Section 2.3.2 we discussed economic phenomena due to which businesses bundle products, often in a multi-enterprise cooperation. A first step in developing a multi-enterprise bundled service offering is the design and assessment of a business model. Such a model shows the actors involved and what they exchange of economic value with each other. Multi-enterprise business models are often very complex, so verbal communication is no longer sufficient to analyze and communicate the whole model. Gordijn & Akkermans (2003*a*) argue that a conceptual modeling approach for such an analysis helps stakeholders reach a better understanding of the business model, and enables them to assess the profitability of suggested business models. They suggest the e^3-*value* method (Gordijn 2002) a conceptual modeling approach for exploring, analyzing and evaluating the economic value perspective of multi-enterprise innovative e-business ideas.

e^3-*value* is a multi-actor approach for developing e-business models, taking into consideration the importance of economic value for all actors involved, and the intertwining of business and technology. When applied to the service industry, an e^3-*value* business model does not provide a logical framework for reasoning about how to bundle services. Such a business model cannot describe in detail the variety and complicate nature of potential service bundles. Nor does it handle inherent dependencies between multiple services, such as ‘service X adds value to service Y’. This information is necessary in order to design feasible service bundles and to point out differences between and redundancies among possible service bundles. Performing

a business analysis of such scenarios requires incorporating such reasoning mechanisms in existing conceptual modeling work, so that the extra information on services is available, to facilitate a complete business model analysis of service bundles. We propose to extend the e^3-*value* method to fill this gap. Both the e^3-*value* method and the extension shall be based on a similar conceptual modeling approach, and the extension will provide the reasoning capacity on the bundling of elementary services into one package.

2.3.5 Degree of Formality

A conceptual modeling approach "formally describes some aspects of the physical and social world around us for purposes of understanding and communication" (Mylopoulos 1992). Software specification methods and techniques use differing levels of formality: informal, semi-formal and formal (Wieringa & Dubois 1998).

Informal techniques use natural language. Though straightforward in human-to-human communication, natural language has several main limitations for the automation of processes, e.g., inherent ambiguity and the difficulty in reasoning with knowledge, required for proving properties of information systems (Ambriola & Gervasi 1997).

Formal techniques are mathematics-based or logics-based techniques where syntax and semantics are defined, and rules to reason with modeled information adhere to the defined semantics. Their employment is time consuming, and they are not well-suited for communication with stakeholders that are not seasoned users of formal specifications. They can best be employed when an envisioned system is too complex to reason without formal support (Wieringa & Dubois 1998).

Semi-formal techniques use diagrams and structured natural language (Wieringa 1998) to structure information, and may offer rules to manipulate, or reason with that information. On the one hand, they have a formal background, making it possible to structure natural language, to define semantics, and to discover errors when dealing with ill-structured information (Wieringa & Dubois 1998). On the other hand, the lack of a mathematical language in these methods results in a lower ability to discover inconsistencies. Their employment is less time consuming, and they are more suitable for interactions between stakeholders, especially those who are not schooled in formal specifications.

Our approach aims at letting business oriented stakeholders describe (model) services. They normally describe their services in natural language, and are mostly unfamiliar with formal specification methods. Hence a high level of formality is not suitable. Yet, the suggested approach should provide the semantics for software-aided reasoning on the design of service bundles out of more elementary services; hence formality is required. In view of the above, we opt for a semi-formal approach.

The ontology that we suggest in the following chapters shall be specified mainly using UML diagrams, and it was also implemented using a Web based knowledge representation (RDFS). We use a formal, logic-based representation to describe constraints in the ontology (see Appendix A), and further explain the ontology using natural language.

2.4 Requirements for a Service Ontology with a Focus on Value Bundling

Having established the need for a (semi)formal description of services to facilitate software-aided reasoning on service bundling, we suggest a service ontology as an answer for this need. In Chapter 3 we present our service ontology, preceded by an introductory discussion on ontologies. In the current section we present requirements for a service ontology that focuses on bundling objects of economic value. The requirements are derived from Section 2.3, where we present the need for such an ontology.

2.4.1 Descriptive Information: the Value of Services

In Section 2.1.2 we defined services as economic activities: activities in which customers and suppliers exchange *value objects*, objects that encapsulate economic value for at least one of the actors involved (Gordijn 2002). Also when customers buy a service, in fact they are not interested in the service itself, but in the benefits – the value – that this service presents for them (Teare 1998). The same principle – in the context of goods – was acknowledged three decades earlier by Lancaster (1966).

Benefits may be tangible (e.g., the possession of an object) or intangible (e.g., a status symbol provides an "experience" benefit; an insurance provides the "capability" to use some service if a predefined situation occurs). In return for these values (benefits) the customer is willing to give some other value, including the price of the service and possibly more (e.g., a commitment to consume a service from a certain supplier for a long period). And hence a service is a business activity, where customers and suppliers exchange value objects.

Owing to this definition of a service, also the bundling of services into a service bundle should be understood as aggregating exchanges of value objects by customers and suppliers. Hence, the exchange of economic values (benefits and costs) should be central to a service ontology in describing services and in configuring (bundling) them. A service ontology should express the economic nature of benefit exchanges between customers and suppliers.

2.4.2 Supplier and Customer Perspectives

We argued in Section 2.3.3 that the realization of e-service offerings presents new challenges, including a supply-side challenge and a demand-side challenge. From a supplier perspective, a service ontology should (1) describe services in a supplier terminology, so that different services can be compared; and (2) provide a mechanism to define business rules for the relations between services. Equally important is the demand-side perspective: a service ontology should describe services in terms of a customer. Customers typically use a different terminology and have a different view on their needs than suppliers (Vasarhelyi & Greenstein 2003). For example, from a supplier's perspective, American Express or MasterCard may consider their customers as consumers of a payment facility in the form of plastic credit cards. Customer may consider the same service as a means to achieve financial security when traveling.

The need for both perspectives is directly related to a main characteristic of services: their intangibility. Goods are tangible and physically observable; both customer and supplier would see the same thing if they observe a good, and the good can be described in unambiguous terminology, including weight, size, shape and more. Services, on the other hand, often have a high degree of intangibility. They are not physical objects that can be observed and described in unambiguous terms based on their physical properties. Instead, they are experienced differently by different customers. They are observed subjectively. Supplier terminology describes how a suppliers wishes to present his service; but because every customer experiences the service differently, this supplier terminology is not suitable for describing the desired service from a customer's perspective. Yet, also the supplier terminology is required, in order to compare the services of different suppliers, to select the most suitable one for a given customer (cheapest, fastest, ...).

Thus, the reasoning process that results in offering a certain set of services to a customer includes knowledge about customer needs (demand-side perspective), about available services (supply-side perspective) and about relationships between the two perspectives. The two perspectives and a transformation between them should be present in a service ontology.

2.4.3 Configurability

As explained in Section 2.3, we propose a service ontology as a means to facilitate the bundling process of services, defined as economic activities. Consequently, a main requirement for a service ontology is to include constructs that support automated reasoning on the service bundling process.

Grönroos (2000) explains that according to the often-used *service package model*, a service is in fact "a package or bundle of different services, tangibles and intangibles, which together form the service".

By way of comparison, a service can be seen as a bundle of small components, that together form a bigger component. Thus, designing a service bundle is a constructive activity. From the knowledge systems literature it is known that such synthetic tasks can be reduced to more tractable tasks under certain assumptions on the knowledge structures of a domain (Top & Akkermans 1994, Stefik 1995, Schreiber et al. 2000). In particular, configuration is a simpler constructive task, where predefined components are configured into a larger, complex component, based on the availability of a set of predefined connections, and associated parameters and constraints (Mittal & Frayman 1989, Löckenhoff & Messer 1994, Gruber et al. 1996).

A configuration task is defined by Mittal & Frayman (1989):

> "Given: (A) a fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at that port, and other structural constraints; (B) some description of the desired configuration; and (C) possibly some criteria for making optimal selections.
> Build: One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connections between the components in the set, or detect inconsistencies in the requirements."

Bearing in mind the similarities between service bundling and component configuration, we require from a service ontology that it provides the means to *configure* individual services into a service bundle. To support this feature, a service ontology has to consider how services can be "connected", and how business rules can serve as constraints on "connecting" services to each other in the bundling – or configuration – process.

A service ontology should thus enable representing the service bundling problem as a component configuration task, as studied in the knowledge systems literature. To achieve this, a service ontology should at least be suitable for (1) describing service components according to the definition of Mittal & Frayman (1989), including a description of relations that represent conditions/constraints for connecting these components, and (2) describing requirements for the bundling (configuration) process. We do not set the third, optional, requirement from the above definition, to provide criteria for optimal selection among solutions. Instead, we are interested in *all* possible solutions, so that business analysts and other domain experts can further analyze all possible bundles. Optimizing criteria may be a later step.

Based on our earlier definition of the term service as an activity of exchanging economic values between customers and suppliers it should be understood that this configuration requirement is different from the configuration of software components, referred to as (web) service composition (Benatallah et al. 2003, Gómez-Pérez et al. 2004, OWL Services Coalition 2004, Paolucci, Sycara & Kawamura 2002, Pires et al. 2002, Sirin et al. 2004, Yang 2003, Martin et al. 2005) (for a discussion on the difference between services and web services, see Section 2.1) or as (web) service configuration (Omelayenko 2005, van Splunter et al. 2003, Jain & Schmidt 1997). Neither is it similar to process configuration, due to the different nature of the items to be configured: activities of economic nature, versus activities of operational nature. In describing a service as an activity of economic nature, one focuses on *what* is offered *by* whom *to* whom. In describing an activity of an operational nature, one focuses on *how* the service (being an economic activity) is realized.

The knowledge engineering literature acknowledges several types of synthesis tasks, including configuration, planning and scheduling (Schreiber et al. 2000). Important differences between configuration and planning are (1) their inputs (planning involves goals that are reached via intermediary sub-goals; configuration does not), (2) their outputs (a configuration task results in an artifact description; planning results in an action plan to achieve a goal), (3) the knowledge they use (configuration requires knowledge of a set of components, planning requires knowledge of a set of actions), (4) planning involves a time-order, while configuration does not (Schreiber et al. 2000). There are two reasons why we discuss configuration, rather than planning. First, in our case we seek to design bundles based on a given set of available services: a set of components, and we do not have goals and sub-goals to work towards. Second, services are described as acts of economic value exchanges, rather than as business processes, where time is required to transform physical, human and information resources into a product. The notion of time is not present in our envisioned model. As a result, also the notion of scheduling is not part of our discussion. In scheduling, the activities of an action plan (the output of planning) are allocated to time slots for execution. This is opposed to research done by the OWL-S community, where web service composition includes web service execution, and the notions of time and planning are therefore dominantly present (Martin et al. 2005).

2.4.4 Graphical Representation

In information science we apply structured and (semi)formal techniques to business topics, which are typically neither well-structured nor well-defined (at least, in the eyes of computer scientists). Business people and other domain experts are an important target group for our work. Moreover, our work depends on a good cooperation with them, so the ability to communicate our ideas to them is crucial. Since this group is not accustomed to formal notations, a different means is required to communicate

our ideas. The use of graphical representations for communication with this target group has been suggested in various contexts, e.g., requirements engineering (Lalioti & Loucopoulos 1994), business process modeling (Oude Luttighuis et al. 2001) conceptual modeling (Parsons & Cole 2005), business value modeling (Gordijn 2002) and IT architectures (Baida 2002).

In view of the above, we require that the service ontology is supported by a graphical syntax, suitable for communication with business people and other domain experts. Also this requirement distinguishes our ontology from work done on semantic web technologies (e.g., RDF, OWL-S) and web services languages (e.g., UDDI, WSDL, SOAP).

In Section 2.3.4 we explained that a service ontology shall be used to perform business analyses together with the e^3-*value* ontology. The latter has a graphical representation, used for communication with domain experts and business people. If both ontologies are to be used together, the service ontology requires a graphical representation as well, preferably one that corresponds to that of the e^3-*value* ontology.

2.5 Product Classification Schemes

So far, the Internet has mainly been used as a channel for selling goods. It is for instance quite common that customers can configure a complex good (e.g., a PC) out of more elementary components and order such a good online. Examples can be found on websites of market leaders such as Dell and Cisco. Such an e-commerce scenario requires a component-based ontology of *goods*, specifically suited for *classification* (to allow customers to find goods) and *configuration* (to facilitate in composing complex goods). Examples of such supporting ontologies are UNSPSC (*UNSPSC website* 2005) and eCl@ss (*eCl@ss website* 2005).

However, from an economic perspective, services grow more and more in importance (World Trade Organization 2003), and will be offered and deployed via the Internet increasingly. With the rise of the service sector, also the notion of 'service bundling' becomes important. Many services are sold as packages, either with other services or as a combination of services and goods (Lovelock 2001, Normann 2001). Together these bundled services (and possibly goods) present the value that a customer seeks (Holbrook 1999, Normann 2001).

The selling of *goods* over the Internet is currently supported by so-called *product classification schemes*. Remember that the term 'product' should embrace goods *and* services. We seek to understand whether these product classification schemes can be used for describing services for the scenario sketched above, or whether they are in fact goods classification schemes.

2.5.1 The Logics Behind Product Classification Schemes

Before starting our discussion on product classifications, it is worthwhile to note the difference between *identification* and *classification*. Identification codes can be used to uniquely identify a product, and consequently to facilitate comparison of the same product among various suppliers, or to enable purchasing management to effectively analyze expenditures (Granada Research 1998). Classification codes, on the other hand, are used to create categories of products, according to some classification criteria (Granada Research 1998). While classification codes make sense for services, the usefulness of service identification codes is doubtful, for no two services are the same (Normann 2001), or are perceived the same, due to subjective factors as the influence of service personnel, company image and customer expectations.

Why Classify

Cunningham et al. (2004) and Lovelock (2001) quote Hunt (1976) concerning the goal of classification schemes:

> "Developing a classification scheme for services... is used for building up theories in research areas and explaining various phenomena."
> "Classification schemes play fundamental roles in the development of a discipline, since they are primary means for organizing phenomena into classes or groups that are amenable to systematic investigation and theory development."

Using a product classification scheme is beneficial for a variety of stakeholders, for different goals:

- Buyers: find all suppliers of a given product or a given category of products, integrate entire process flows, analyze expenditures (Granada Research 1998).
- Suppliers: maintain product catalogues, improve productivity by minimizing ordering errors, easy interface for *all* customers, create market awareness among customers (Granada Research 1998).
- Market research community: determine market share and shifts in consumer demand (Ambler 1998).
- Trade analysts: compare imports and exports to domestic production, compare statistics across countries, construct price indices (Ambler 1998, Mohr 2003*a*).

How to Classify

Whether used by a company (e.g., for benchmarking) or by a national or international statistics organization (e.g., for comparing import and export of certain goods among countries), a product classification scheme has to be hierarchical in order to facilitate analyses of (1) a product in relation to comparable products, (2) product families or (3) high-level classes of products. A hierarchical scheme also facilitates the search of similar or related products, allowing a company to dynamically use differing levels of abstraction for marketing, search or comparison of products (Granada Research 1998).

Two perspectives are possible in the design of a product classification scheme: supply-side and demand-side. A supply-side perspective focuses on how products are produced, while a demand-side perspective uses the markets of a product as a starting point (Donnelly 1999). As we show in the next section, most major product classification schemes use a supplier perspective.

Ambler (1998) presents an analysis of four approaches for classification schemes. The analysis was performed by the US Economic Policy Committee (ECPC) (*ECPC Issue Paper 1* 1993):

1. **Demand-based product classification**: products that are used together or that define a market shall be classified together. Possible classification criteria are:

   - Substitute products belong together. Implementing this method has proven to be difficult.
   - Products belong together if their prices move together.
   - Products belong together if they are used together.
   - A product belongs to a class if the demand for that product depends only on the prices of products within the class and possibly on consumer income.
   - Marketing relationship: products that are sold through the same channels are classified together.

   A demand-based approach is beneficial mainly for doing market studies and analyses.

2. **Intrinsic nature or physical characteristics of the product**: classification criteria are (1) the material of which the goods are made, or (2) the degree of processing of the product. This approach is focused on goods, as services mostly cannot be described in terms of physical characteristics. A supply-side oriented classification based on physical properties of the product is beneficial mainly for analyses of foreign trade on the domestic market.

3. **Industry of origin**: a product is classified together with other products that are produced by the same industry. Many products, however, may be produced and sold by multiple industries, resulting in the conclusion that creating a supply-side classification scheme based on industry of origin is practically infeasible.

4. **List of products**: an exhaustive list of products, ordered in some described manner (e.g., alphabetically) so that users can re-order the list based on their needs. Although it gives users a high level of flexibility, this approach does not allow the comparison of data between organizations that re-order the list in different ways.

2.5.2 Existing Product Classification Schemes

In this section we discuss the usage of major existing product classifications.

CPC

The main purpose of the Central Product Classification (CPC) (United Nations 2002) is to enable international comparisons of economic statistics dealing with products, through harmonization among various fields of economic and related statistics. CPC was the first international classification scheme to cover also the outputs of service industries; it includes a hierarchy of products, including "transportable goods, non-transportable goods (e.g., bridges) and services" (United Nations 2002). The designers of CPC did not consider it to be relevant whether a product is a good or a service, since officially CPC was to be used to classify anything that entities may exchange in an economic activity. Nonetheless, CPC's classification criteria reflect the main focus of the scheme: goods, and not services. The main classification criterion in the CPC five-levels hierarchy (Section, Division, Group, Class, Subclass, each specified by one digit) was physical properties and the intrinsic nature of the products. The industry of origin was considered a main, yet second, criterion in classification categories into a higher-level category (Ambler 1998).

UNSPSC

In 1999 UNSPSC became the result of a merger between the United Nation's Common Coding System (UNCCS), itself based on the United Nations Common Procurement Code, and Dun & Bradstreet's Standard Product and Service Codes (SPSC). The hierarchical goods and services classification scheme contains the following five levels (Granada Research 1998), each specified by two digits: (1) Segment (the logical aggregation of families for analytical purposes); (2) Family (a commonly recog-

nized group of inter-related commodity categories); (3) Class (a group of commodities sharing a common use or function); (4) Commodity (a group of substitutable goods or services); and (5) Business function (The function performed by an organization in support of the commodity). The main goals of UNSPSC are to enable product discovery (i.e., procurement activities), expenditure analysis and increasing product awareness among prospective customers who search products (i.e., marketing activities) (Granada Research 1998). Although not literally stated but yet visible from UNSPCS' goals, the UNSPSC is a supply-side scheme. It includes segments based on the product category: *paper materials*, *fuels* and *chemicals*, but also various industries as *building and construction*, *environmental services* and *healthcare*.

eCl@ss

The eCl@ss product classification scheme is an initiative of several German industries. It includes descriptions for goods and services, to support highly automated virtual marketplaces, sales, procurement, engineering, plant maintenance and warehouse management (eCl@ss 2000). It includes four levels, specified by two digits each: Segment, Main group, Group and Commodity class (altogether referred to as 'Material Class Hierarchy'). It includes also a keyword system, so that users can search for classes. At the lowest level of the hierarchy, commodities are described by attributes. The classification of elements adheres to three considerations (eCl@ss 2000): (1) a parent class is equal to all its children, and the children are subsets of the parent; (2) classes are formed according to market segments, which in turn are formed based on raw materials (supply-based), based on used technology (supply-based), or – in third place only – based on what a customer or application needs (demand-based); and (3) the classification scheme should be usable for economists as well as technologists (and yet eCl@ss acknowledges the fact that it is impossible to satisfy both groups at the same time, so both will have to compromise).

NAICS and NAPCS

NAICS, the North American Industry Classification System, is an industry classification scheme used by the USA, Canada and Mexico to facilitate the comparison of industrial production statistics among the three countries. NAICS' categories are first and foremost industry-driven (supply-oriented): establishments that use the same production processes to produce a good or service should be classified together (Ambler 1998). Within an industry, the classification of products should be market-oriented (Mohr 2003*a*). This results in a four-levels hierarchy: Economic sector, Economic subsector, Industry group and Industry.

In 1999 a joint effort was launched by the USA, Canada and Mexico, to develop

a demand-based product classification scheme. The North American Product Classification System (NAPCS) would complement the NAICS and be compatible with it. NAPCS would include a product classification for all products produced by all NAICS industries (Mohr & Russell 2001), after the NAICS has been elaborated with service industries in 2002. Although based on NAICS industries, NAPCS would be demand-based, rather than supply-based, to support "many purposes, including studies of market shares and the demand for goods and services domestically consumed and internationally traded" (Mohr & Russell 2001). NAPCS is still under development. The final classification criteria are promised to be demand-based, and to reflect how products are used; common products will carry a common title, definition, and product code across all industries that produce it (Mohr & Russell 2001). As Mohr (1999) warns, creating a demand-based classification scheme is not a trivial task, as various users would require different classifications, to generate their desired statistics. For studies of monopoly power, substitutes or products that are used together should be classified into one category (class), while for other market analyses it is desirable that all close complements to a given product are classified together.

NAPCS will contain 45 product groups: groups 1-24 are mostly consumed by households/persons, whereas groups 25-45 are first and foremost used by businesses as inputs for production processes (but may also be consumed by persons/households) (Mohr 2003*b*). Due to the increasing importance of service industries and to the fact that so far classification schemes concentrated mostly on goods and not on services, NAPCS will put an emphasis on products of service industries.

RosettaNet

RosettaNet is a non-profit standards consortium that develops and seeks to promote deployment of Internet-based standards for the support of B2B communications. More than 500 organizations participate in RosettaNet, representing a variety of industries: Electronic Components (EC), Computer & Consumer Electronics (CCE), Logistics (LG), Semiconductor Manufacturing (SM), Solution Provider (SP) and Telecommunications (TC) (*RosettaNet website* 2005). Unlike other classification schemes, RosettaNet does not use numbers to identify products, but instead it uses names, and refers to their UNSPSC code. The hierarchy is very minimal, and includes only two layers: RN Category (for a class of products) and RN Product (for a specific product). RosettaNet consists of 14 categories and around 150 products, and focuses mainly on electronic equipment (Corcho & Gómez-Pérez 2001).

2.5.3 Product Classification Schemes: Analysis

A classification is always biased towards the goal that it serves. In other words, a classification must provide specific insights into the domain being classified; elements of

Table 2.3: Product Classification Schemes: Comparison

	Goal	**Design perspective**	**Product description possible?**	**Classification criteria**	**Focus: goods or services**
CPC	Compare economic statistics	Supply	No	Physical properties; industry of origin	Goods
NAICS	Compare industrial production statistics among countries	Supply	No	Production process	Goods
NAPCS	Statistics: market share, international trade and more. Planned to be ready for use in 2007.	Demand	Presumably not; however the scheme is not yet ready	Market (households, businesses)	Services and goods
UNSPSC	Facilitate internal and external business activities (procurement, marketing, financial analyses etc) through electronic catalogues	Supply	No	Product category	Goods
eCl@ss	Facilitate internal and external business activities through electronic catalogues	Mainly supply-based. Only the third classification criterion is demand-based.	Attributes are available at the lowest level of the hierarchy	Raw materials of which products are made; technology used to produce the product; common usage	Goods
RosettaNet	B2B communications	Supply	No	Product category	Goods

the domain are therefore classified in such a way that the specific desired insights can be gained. Different classifications of the same domain enable gaining different insights into the domain. This principle was acknowledged by Lovelock (1983) when he offered a new, multi-view approach to service classification, and is important in understanding why existing product classification schemes do not serve us well for service bundling.

Table 2.3 presents a comparison between the above product classification schemes. As can be seen, existing classification schemes are designed either to support statistical analyses (CPC, NAICS/NAPCS) or to support e-business activities (UNSPSC, eCl@ss, RosettaNet).

Schemes for statistical analyses

Classification schemes belonging to this group (CPC, NAICS, NAPCS) are used to compare statistics on trade between various countries (Ambler 1998), regarding classes of products that the classification scheme identifies. An example analysis would be "what is the import/export balance between the USA and Canada for uranium and thorium ores (CPC version 1.1 Section 1, Division 13)?". Such analyses share the characteristic that information needs on products are specified on the abstraction level of Section, Division, Group, Class or Subclass (the hierarchical levels of CPC), so that it is not possible to relate products that belong to differing classes on the same hierarchical level or on different levels; this is not required for the goal for which these schemes were designed.

Schemes for the support of e-business activities

UNSPSC, RosettaNet and eCl@ss were designed to facilitate electronic business, with the emphasis on B2B activities. Our proposal for a service ontology also intends to facilitate electronic business, but not less emphasis is put on B2C activities. This difference has implications on the required terminology for facilitating the activities, and on the use of communication standards as the RosettaNet standards.

In B2B activities customers and suppliers are businesses; they are more likely to share a similar vocabulary and the same perspective on services, and they understand that they need to comply with communication standards to employ cross organizational information systems effectively. Consequently, they understand that it is in their interest to use communication standards.

B2C activities, on the other hand, involve end-user customers: people, rather than businesses. Consumers typically have a different perspective on consumption than businesses do: customers seek for the subjective benefits of a service, while businesses tend to describe their services in objective terminology, often in terms of provided functionality rather than in terms of customer benefits. Consumer perspective and vocabulary are different from that of suppliers. But unlike businesses, consumers do not need to adjust their vocabulary and way of communicating to that of suppliers. In an economy that becomes more demand-oriented (Normann 2001), it is the

supplier who changes to fit customer needs, and not the other way around. It is the supplier who tries to convince consumers that he can deliver the benefits that consumers seek. It is the supplier who has to convince consumers that he delivers value. Software-aided reasoning on this level requires an understanding of the benefits that a service provides to customers, as customers perceive it. For this reason, supporting B2C activities should involve not only a supplier perspective, but also a customer perspective. The same reasoning can also explain why existing communication standards are not likely to be of help for end-user customers. These standards describe a supplier perspective on product offerings, and fail to describe the consumer benefits of consuming these offerings.

To understand whether schemes of both types can serve for service bundling, let us examine whether they satisfy the earlier described requirements for a service ontology.

Discussion on the requirement to describe services from a business value perspective

An early, very basic question to be asked is whether existing classification schemes describe services at all. Although the answer for this question is positive, NAPCS will be the first and only product classification scheme that in fact focuses on services at least as much as on goods. Officially also other schemes support services, but their goods-biased classification criteria leave no doubt about their strong goods-orientation. CPC and eCl@ss, for example, classify products first and foremost based on their physical properties and raw materials, respectively. Services, however, can hardly be described by these criteria.

But let us assume, for the sake of discussion, that existing schemes *do* focus on services. Are they then sufficient to describe the *value* of services, so that services can be selected based on the value that they provide to customers? Let us look at example services in these schemes. CPC version 1.1 Class 8525 (Guard services) includes the following services: security patrol services, security guard services, bodyguard services, watch-dog services, parking control services and access control services. As is often done for goods, this is an attempt to describe services in "unambiguous" functionality-related terms. However, services are not tangible like goods, and hence this description is not enough. No mechanism is provided to describe what, in fact, the customer receives (what are 'parking control services', and which intangible (experience-related) or tangible benefits do these services provide to a customer?). Furthermore, no mechanism is provided to describe quality criteria or other properties of a service, since this is not required for the goal of generating trade statistics. Similar things can be said about NAPCS. Group 16 (travel and lodging products for persons, leisure and business) of NAPCS includes, among others, the following

leaf-products: motor homes, travel trailers, and campers; air travel; intercity rail and related travel; intercity and chartered bus travel; and cruise ship travel. Also these are merely categories that describe a functionality, rather than the benefits of a service, and certainly no mechanism is available for describing properties of such services.

Descriptive information is of greater importance for classification schemes for the support of e-business activities. Indeed, eCl@ss provides a set of attributes for describing products. Cologne Institute of Business Research (2000) shows that (1) eCl@ss attributes are process-oriented, in accordance with the eCl@ss vision to "enable highly automated business processes" (eCl@ss 2000) and with the importance of supporting ERP systems, as reflected in the eCl@ss requirements; (2) eCl@ss attributes focus on describing goods, rather than services. Since eCl@ss, RosettaNet and UNSPSC share a supply-side perspective, the descriptive information that they provide is good enough to provide an unambiguous description of goods, in terms of their functionality and physical properties. This is needed when businesses want to sell goods, and their potential customers know exactly which goods they require. As argued before, end-user customers, on the other hand, often do not use the same vocabulary as suppliers, and they require a service description that reflects the value of services, rather than just the functionality of a service, as a supplier describes it. Providing this information is not facilitated by classification schemes.

Discussion on the requirement of two perspectives

As can be seen in Table 2.3, all classification schemes we examined except for NAPCS have a supply perspective. NAPCS is promised to have a demand perspective. Yet, NAPCS is not aimed at reasoning about customer needs and how they can be satisfied by available services, or service bundles. Instead, NAPCS is aimed at generating statistics about markets. This market orientation is indeed a demand-side perspective, but a different one from what we need here; it requires different information and knowledge, as described in the above discussion concerning descriptive information on services. In other words, NAPCS is the only classification scheme in this group that represents a demand-side perspective; but also this perspective is targeted at describing markets, rather than describing services from a customer's point of view.

Discussion on the configurability requirement

The configurability requirement of the service ontology states that a service ontology should at least be suitable for describing service components and relations that represent conditions/constraints for relating these components to each other (see Section 2.4.3). Both the component-like structure and inter-relating services are two

principles that schemes for statistical analyses do not require, since their use does not require "building bigger products" out of smaller ones, and because these schemes are used for the analysis of classes of products that are related according to specific, predefined criteria (the classification scheme's classification criteria), rather than any other criteria. Consequently, it is not required to enable users of the scheme to define other relations between products.

For example, NAICS uses the production process as a classification criterion, implying that NAICS can be used to relate products that share the same (or similar) production process(es), but other relations are not supported. CPC, on the other hand, uses the physical properties of a product as the main classification criterion. Consequently, it is possible to relate products with equal or similar intrinsic physical characteristics. In fact, the classification criteria are the only means for relating products in a classification scheme. This is too limiting, since for various reasons one may need to identify different relations. For example, one may wish to search products of a certain type (this is currently supported by classification schemes), but one may also want to identify inherent relations between products. One such relation (substitution) is claimed to be dealt with by the future NAPCS (Donnelly 1999), but other relations such as the disjoint/excluding relation are not being dealt with. These relations are important for understanding customer behavior and relations between suppliers of various products, and for defining compound products.

Classification schemes for the support of e-business activities may be more concerned with a component-like structure than schemes for statistical analyses are. And indeed, in a discussion on attributes to describe eCl@ss references, Cologne Institute of Business Research (2000) claims that "characteristics and sets of attributes for many different *components* and subgroups can be defined" by using eCl@ss. Yet, the term 'component' in eCl@ss refers mainly to a product description (product unique ID and attributes as format, data type, material number, place and plant where it was manufactured, life cycle data etc.), and not to a 'component' as defined in configuration theory (Mittal & Frayman 1989, Löckenhoff & Messer 1994, Gruber et al. 1996). The designers of UNSPSC used the idea of combining products to what is referred to as "a contractible group", a single, self-contained family of products (Granada Research 1998). Unlike the requirement for configurability of possibly unrelated products, in UNSPSC the products to be grouped ('configured' into a single offering) are predefined based on a given business rule: a contractible group includes a number of products that "the buying enterprise" – note the strong B2B focus – "can approach to negotiate a single point of supply" so that "the enterprise can attain preferential treatment (including price discounts) in the contract" (Granada Research 1998).

The lack of flexibility in how current classification schemes support relations between products is paradoxically inherent to their biggest strength: being hierarchical classifications. A hierarchical classification is about classifying elements based on some

predefined criteria, and leaves no space for other classifications, or other relations. Consequently, when a variety of relations between products needs to be dealt with, a traditional product classification scheme does not suffice. The hierarchical structure of classification schemes is required for comparing prices of the same product among various suppliers, or for the derivation of statistics on groups of products. These are indeed goals of all earlier discussed product classification schemes. But in a reality where products (mostly services) are dynamically combined based on customer needs, and where product descriptions must demonstrate the benefits products pose to a customer (Normann 2001), the hierarchical structure does not provide enough flexibility.

Discussion on the requirement of graphical representation

A discussion on a graphical syntax for existing classification schemes is superfluous, since these schemes are merely a hierarchy, with no other conceptual model, process or structure behind them, except for their inherent tree-structure. The only relation in existing classification schemes is "A is subclass of B" (with the inverse: "B is superclass of A"). Neither is a graphical syntax required for their goal.

2.6 Service Classification Schemes

A broad consensus exists among service management and marketing researchers, emphasizing characteristics of services that differ inherently from those of goods: intangibility of services (vs. tangible goods), heterogeneity (non standardization), inseparability (of service production and service consumption), perishability (vs. goods that can be stored) and more (Grönroos 2000, Kasper et al. 1999, Lovelock 2001, Zeithaml et al. 1990). This line of research resulted in a number of service classification schemes that emphasize the differences between services and goods (e.g., Shostack (1977)) and are based on the industry of the service provider (Kotler 1980), as is customary in classifying goods as well.

2.6.1 Existing Service Classification Schemes

As early as 1983, Lovelock (1983) published a review of existing service classification schemes. Two decennia later Cunningham et al. (2004) published a similar review, revealing that in spite of the limitations of classifications from the 1970's and 1980's (Hill 1977, Shostack 1977, Kotler 1980, Lovelock 1983), they are still broadly being used and referred to. Various authors proposed their own classification schemes, incorporating a number of classification criteria (or: dimensions) in each scheme. Prominent examples are shortly discussed here.

Table 2.4: Service classification matrix (Hill, 1977)

		Services affecting goods		Services affecting persons	
		Permanent	Transitory	Permanent	Transitory
Physical changes	Reversible	×	×	×	×
	Irreversible	×		×	
Mental changes	Reversible			×	×
	Irreversible			×	

Hill (1977) suggested a matrix, resulting in nine groups of services (see Table 2.4), based on the following criteria:

- Does the service affect goods or persons?
- Does the service provide a permanent or a temporary change (to goods/persons)?
- Is the effect of the change reversible or not?
- Is the effect mental or physical?

This classification focuses on the nature of benefits of a service, and was created for economic analyses. A fifth criterion is given but not included in the matrix: a distinction between individual and collective services.

Although a variety of service classification schemes preceded him, Lovelock (1983) was a pioneer in the sense that he suggested a different approach to service classification. He suggested "to group services other than by current industry classifications", namely by relevant marketing characteristics. Instead of offering *one* classification scheme, he offered to classify services in five different ways, based on five different marketing characteristics of services. Every classification would offer different marketing insights. The service marketing characteristics, serving as classification criteria, are:

- *The nature of the service act.* Two sub-criteria are used here: (1) services directed at people vs. services directed at things, and (2) is the act tangible or intangible in nature? These questions result in a four-groups classification

scheme. The scheme helps understand whether a customer needs to be physically or mentally present during service delivery.

- *The type of relationship that the service organization has with its customers*. Two sub-criteria are used here: (1) is there a formal ("membership") relationship between customer and supplier or not? and (2) is service delivery continuous, or does it take place at discrete intervals? Once again, these questions result in a four-groups classification scheme. The scheme, praised for its simplicity, helps understand which marketing communication tools can be used effectively and how strong the relationship between the customer and the supplier is.

- *The amount of room there is for customization and judgment*. Two sub-criteria are used here: (1) to which degree can the service be customized (high/low); and (2) to which degree does customer contact personnel exercise judgment in meeting individual customer needs (high/low). Also this classification scheme includes four groups.

- *The nature of demand and supply for the service*. Two sub-criteria are used here: (1) extent of demand fluctuations over time (wide vs. narrow), and (2) extent to which supply is delayed at peak demand (demand can be met vs. demand exceeds capacity). Once again, these questions result in a four-groups classification scheme, used for managing demand.

- *Service delivery*. Two sub-criteria are used here: (1) number of service outlets (one/multiple); and (2) nature of interaction between customer and supplier (customer goes to the supplier, supplier comes to the customer, or transaction at arm's length (e.g., via the Internet or telephone)). These criteria result in a classification scheme with six groups, helping understand distribution issues.

In their study, Cunningham et al. (2004) selected a set of eleven broadly-used managerial classification dimensions, and investigated how they are perceived by customers. The result is a demand-side service classification scheme, as opposed to all other supply-side schemes. Their criteria are the extent to which the customer feels:

1. "The level of physical product component is high or low.
2. The level of the customer-employee contact is low or high.
3. The production and consumption of a service is inseparable or separable.
4. How risky it would be to choose a provider.
5. The switching to a new provider is easy or difficult.

6. A service is performed on a person or on a tangible object.
7. A service exhibits a formal or no formal relationship between the service provider and the customer.
8. The delivery of a service is continuous or involves discrete transactions.
9. The customization of a service is high or low.
10. The contact person exercises high or low levels of judgment when making service provision decisions.
11. The degree to which the convenience level of obtaining a service is high or low."

Their analysis shows that two service dimensions account for 78-82 percent of the total variance in service perceptions and classifications by customers, leading to the conclusion that these two dimensions are of greatest importance for service classification: (1) personalization versus standardization of the service, and (2) presence of goods as part of the service.

2.6.2 Service Classification Schemes: Analysis

In spite of the shared understanding of what services are within the service management and marketing community, there is no common opinion on how to classify services (Kasper et al. 1999). Traditional service classification schemes from the 1970's and 1980's have several drawbacks. First, many schemes use a small number of classification dimensions, failing to cover the broad scope of differences between one service and another (Cunningham et al. 2004). Second, they have been designed from a supply perspective only (as opposed to the more recent demand-side oriented work of Cunningham et al. (2004), described above). Third, while various authors (Rathmell 1966, Kasper et al. 1999) acknowledge the fact that classification dimensions should be viewed on a continuum rather than in a discrete way, classification schemes opt for the discrete approach, as can be seen in the classifications discussed in Section 2.6.1 and in multiple other classifications. From our point of view, all these classification schemes share a fourth drawback, namely their goal. They were designed to be used for economic analyses by marketing departments, to gain strategic managerial insights into service marketing. As we will see, this makes them unsuitable for our case.

For readers who are not familiar with the literature on service classification, it is worth noting that research on service classification has been performed by service marketing researchers, published in marketing journals and textbooks, and used to

Table 2.5: Existing service classification schemes versus service ontology for service bundling

	Service classification schemes for economic analyses	**Service ontology for service bundling**
Usage	*Divide* whole spectrum of existing services into smaller groups	*Combine* services into groups
Classification rules	*Global rules* (hold for the whole service industry)	*Company- and domain-specific* business rules
Nature of classification rules	Classification criteria that *differentiate* one service from another	Any type of dependency between services (e.g., difference, similarity)
Level of reasoning	*Classes* of services (e.g., insurance services)	*Instances* of services (e.g., ABN-Amro private unemployment insurance)

gain managerial marketing insights. Dumas et al. (2001) were right to note that this work from the field of service marketing does not explicitly take into account service automation and service composition. Service automation was still at very early stages when these classification schemes were developed, and service composition was simply not the goal of these schemes.

Having been designed for marketing goals, existing classification schemes differ substantially from the envisioned service ontology for service bundling. Important conceptual differences are captured in Table 2.5. Most important is the level of abstraction: classification schemes facilitate reasoning on the level of classes of services, while the envisioned service ontology will enable reasoning on the level of service instances, to design concrete bundles of service instances.

Discussion on the requirement to describe services from a business value perspective

Since service classification schemes describe *classes* of services, they do not provide any descriptive information on the service instance level. Neither is there any mechanism for descriptive information on the class level, as (1) this is not required for the goal of service classifications, and (2) classes are often too generic for such a descrip-

tion to make sense. A remark to be made is that Hill's classification (Hill 1977) in fact uses the nature of the benefits – the value – of services as classification criteria. Yet, also this scheme does not provide information on the actual value of a service instance, but information on the nature of value (affecting people or things; permanent or temporary, reversible or not; mental of physical effects) of a *class* of services.

Discussion on the requirement of two perspectives

In their review of existing services classifications Cunningham et al. (2004) maintain that a good service classification scheme "should be based on consumers' perceptions" – i.e., use a demand-side perspective – "because it is used in explaining and understanding their behaviors about services", while many of the existing classification schemes were designed from a supplier perspective. They offer their own classification, based on customer perception of services, i.e., demand-side. Yet, a customer perspective is also available in Hill's classification (Hill 1977), because it centers around the benefits that services deliver to customers. Most other schemes indeed use supplier-oriented classification criteria. For example services vs. goods, industry, consumer services vs. industrial services and the service delivery process. None of the existing classification schemes uses both perspectives.

Discussion on the configurability requirement

Similarly to product classification schemes for statistical analyses, also service classification schemes are not meant to support the design of complex services out of more elementary services. They are meant to demonstrate which classes of services show predefined similarities, reflecting marketing-related similarities between these services. It should be noted though that some service classifications show greater flexibility than product classifications, because they use a matrix-like structure, compared to the one dimensional structure of product classifications.

As described before, the configurability requirement of the service ontology states that a service ontology should at least be suitable for describing service components and relations that represent conditions/constraints for relating these components to each other (see Section 2.4.3). Service classification schemes do not provide a mechanism to describe a service, except for the classification criteria of the class where the service is classified (Cunningham et al. 2004). Nor do they provide a mechanism to define relations between services (independent of their belonging to a category), since defining such relations is not required for performing the economic analyses for which they were designed.

Discussion on the requirement of graphical representation

Similarly to our discussion on product classification schemes, also here the discussion on a graphical representation is superfluous, because service classification schemes do not include any structure or relations that can be modeled graphically. They merely separate the spectrum of services into classes that are said to differ from one another, and a graphical syntax is not required for their goal.

2.7 Concluding Outlook

In the previous sections we discussed an envisioned service ontology for modeling domain knowledge from a business value perspective. We require from the ontology that:

- It describes the value exchange that services encapsulate.
- It describes services from a customer perspective and from a supplier perspective.
- It facilitates representing the service bundling task as a configuration task.
- It has a graphical representation, next to a computer-processable one.

In the rest of this thesis we present a service ontology that satisfies these requirements.

Our ontology describes services through the benefits that they deliver to customers (e.g., capabilities, experiences, physical goods and more) and the benefits that must be provided by customers who wish to obtain the service (typically a fee, but other possibilities exist as well). These benefits are the economic values that customers and suppliers exchange, in accordance with the principle of economic reciprocity.

We explicitly separate the supplier perspective on services from the customer perspective. The supplier perspective is used to describe services such that actual service instances can be discovered, compared and composed into service bundles. The customer perspective is used to ensure that service bundles actually satisfy customer needs. Only if the logic of both perspectives is captured and modeled, can we ensure that solutions make sense for both parties involved in the value exchange.

The supplier description of services is used for the actual design of service bundles. We consider services to be *components*, building blocks for service bundles. We use the supplier description of service components together with a configuration ontology to represent the service bundling task as a traditional configuration task. As a result,

service bundling can be seen as a configuration task, although the objects that we configure are mostly intangibles, as opposed to traditional configuration of tangibles.

Ontologies as formal conceptualizations aim to bridge human and computer understanding. We show how this is achieved by combining two different representations of the same knowledge into one software tool . On the one hand, a computer-processable representation of the ontology facilitates reasoning by software. On the other hand, we present also a visualization of our ontology, to enhance communication with domain experts and business analysts. Consequently, humans can visually model domain knowledge, with which software can then reason.

Part II

Service Ontology: Theory and Implementation

Chapter 3

A Service Ontology with Demand and Supply Perspectives

***Note**: This chapter presents the Serviguration service ontology, the main output of our research. An earlier version of Section 3.3 was published as a paper in the proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004) (Baida, Gordijn, Sæle, Morch & Akkermans 2004) and in an article in the IEEE Intelligent Systems magazine (Akkermans, Baida, Gordijn, Peña, Altuna & Laresgoiti 2004). Sections 3.4 and 3.5 were published in the proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005) (Baida, Gordijn, Sæle, Akkermans & Morch 2005).*

In Chapter 2 we presented the need for a semi-formal approach to service bundling. We argued for a means to reason systematically about services, interpreted as economic activities in which customers and suppliers exchange objects of economic value. We also argued that it is necessary to conceptualize domain knowledge on services and make this knowledge computer-processable, so that software can perform the service bundling task. This is required for e-service scenarios, and can serve as an important tool in conducting business analysis studies for networked enterprises.

In this chapter we present our solution for this need: a service ontology that can fill in these gaps. We start with a short discussion on ontologies in general and on the use of ontologies in a context of business research.

3.1 What is an Ontology?

The term 'ontology' stems from the Greek "on", meaning *being* ("ontos" means *of the being*) and "logos", meaning *language* or *reason*. In the discipline of philosophy it refers to "the science of being" (Roche 2003). Computer scientists and artificial intelligence researchers and practitioners have more recently started using the same term to express a shared understanding (within a community) of what is believed to exist. An 'ontology' was defined by Gruber (1995) as "an explicit specification of a conceptualization". This definition was somewhat altered by Borst (1997) who defined 'ontology' as "a formal specification of a shared conceptualization". Studer et al. (1998) explained both these definitions:

> "A 'conceptualisation' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. For example, in medical domains, the concepts are diseases and symptoms, the relations between them are causal and a constraint is that a disease cannot cause itself. 'Formal' refers to the fact that the ontology should be machine readable, which excludes natural language. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group".

Sharing is a key issue in our service ontology. A shared understanding of services is required to enable customers and suppliers buy and sell services via the Internet, to enable joint offerings of various suppliers over the Web, and to enable business analysts conduct a business analysis for networked enterprises. The conceptualization has to be shared in order to support communication between humans, computers and enterprises. This was summarized by Gruber (2004): "every ontology is a treaty – a social agreement – among people with some common motive in sharing".

3.2 Positioning the *Serviguration* Service Ontology

Ontologies facilitate sharing and re-use of knowledge by capturing the intended meaning of concepts and relations in a domain. They are currently a main topic of research within computer science and artificial intelligence faculties, where research is done on ontology engineering. Since ontologies are seen as automation enablers, research on ontologies often focuses on *how* business transactions can be executed by software. Reasoning about the actual business value of these transactions is often neglected, as this topic has traditionally been studied in business schools rather than in faculties of (exact) sciences.

3.2.1 Using Ontologies in a Business Context

Yet, the use of ontologies within business schools is limited. We queried the database of *Emerald Group Publishing* for articles containing the word ontology as part of the title, keywords or abstract. Emerald is a publisher of academic journals for business disciplines, with journals as *European Journal of Marketing*, *International Marketing Review*, *Journal of Services Marketing*, *International Journal of Service Industry Management* and more. We chose for Emerald because many papers relevant to our research are published in the *International Journal of Service Industry Management* and in the *Journal of Services Marketing*. The query, performed in April 2005, resulted in only 24 articles. In the vast majority of these articles, the term 'ontology' was used in what Heylighen (2001) refers to as the "original philosophical meaning" thereof, as opposed to the definition of Gruber (1995). Only five of the 24 articles describe and use a domain ontology. Three of the five articles describe how a domain ontology was used in the development of an information system or website. Only two of the articles (Martin & Marion 2005, Scozzi & Garavelli 2005) – both from 2005 – discuss the use of ontologies for a domain analysis, independent of information systems development, and only one of these articles (Scozzi & Garavelli 2005) discusses the use of a domain ontology for business analysis or business development. The low number of research efforts involving ontologies for business analysis/design/development was attributed by Scozzi & Garavelli (2005) to the fact that these research issues are "highly unstructured and characterized by difficult-to-forecast activities linked by reciprocal rather then sequential dependencies", so that structured techniques were thought to be inappropriate. This belief has been changing in recent years, as several authors have been arguing in favor of structured reasoning techniques, using ontologies, when adopting the viewpoint of business analysts and marketeers.

In their *Business Process Handbook* project, Malone et al. (1999) describe an ontological business process modeling approach for business process redesign and inventing new organizational processes. They propose a complex ontology of more than 3700 concepts (Klein & Dallarocas 1999) , and describe its use in redesigning the hiring process of a firm. The approach, using knowledge management techniques for a business modeling task, was described by the authors as "novel" in 1999 (Malone et al. 1999).

Presley et al. (2001) discuss the need for modeling business processes. They show how structured modeling approaches such as object-oriented modeling and ontologies can help engineer and improve business processes within a virtual enterprise. Their work does not use an own virtual enterprise ontology, but uses existing ontologies as the AIAI enterprise ontology (Uschold et al. 1998) to argue for a structured modeling approach for business process modeling.

Osterwalder & Pigneur (2002) introduced an e-business modeling ontology to sup-

port a rigorous definition of issues that influence e-business concerns and their interdependencies in a company business model. They argue that such an approach can help companies "understand, communicate and share, change, measure, simulate and learn more about the different aspects of e-business in their firm".

Gordijn (2002) proposed an ontology to describe business models, to explore and understand business ideas and evaluate their potential profitability. His ontology provides reasoning capacity for understanding the intricacies of multi-actor business models. He argues that the use of such a structured approach results in clarity about value propositions of all involved actors, while the lack of this clarity has been a cause for failure of e-commerce initiatives in the past.

Scozzi & Garavelli (2005) studied the use of business modeling techniques (BMTs) that support the innovation development process within small and medium enterprises (SMEs) from several perspectives, e.g., sequence of tasks, decisions that evolve over time, strategic process, political process, and communication and information flow. They classified BMTs based on the kind of ontology that the BMT uses: activities, states, decision criteria and decision variables, roles, concepts and data flow. Next, they analyzed which of the BMT classes is most suitable for analysis of each of the perspectives. Their research shows that models and structured analysis techniques can be a helpful tool in creating strategies, in reasoning, in gaining insights and in communication.

Our research continues the line of the above and other authors, arguing for a structured modeling approach of business issues, that are typically unstructured and ill-defined, in terms of computation. The service ontology we present in this thesis captures domain knowledge to enable reasoning processes that are currently performed in the minds of service personnel. In the studies we present in Chapters 7 and 8 we show how the use of our ontology facilitates a systematic reasoning process with business knowledge. Domain experts who used our ontology declared that the use of our ontology helped them gain new insights into their own domains.

3.2.2 Viewpoints on E-Service Provisioning

Gordijn & Akkermans (2001) distinguish between three viewpoints in the design of e-business activities: a business value viewpoint, a business process viewpoint and an information system viewpoint. The business value viewpoint focuses on ways to create, distribute and consume economic value. Relevant stakeholders are CxO's (e.g., CEO, CFO), marketeers and customers. The business process viewpoint focuses on business processes, through which value propositions are put into operation. It focuses on ownership of these processes. Relevant stakeholders are those who are responsible for the design and execution of operational processes (e.g., operational management). The information system viewpoint represents the information systems

that enable and support the business processes. Relevant stakeholders are those who are responsible for development and exploitation of information technology, typically employees of IT-departments. We adopt these three viewpoints, and add a fourth one: the computer science / artificial intelligence viewpoint. It focuses on upper level ontologies and domain ontologies. These are used as a basis for building information systems. Main stakeholders are knowledge engineers.

In Section 2.3 we argued that software-aided support for service bundling is required for the realization of e-service offerings, and for business analyses of service offerings involving networked enterprises. The first of these two contexts fits well into the framework of Gordijn & Akkermans (2001), since e-services are a type of e-business activities. The second context is not limited to e-services, but includes also traditional services. However, since business and IT are nowadays strongly intertwined, and most services require some support of IT, we can safely adopt the three earlier mentioned viewpoints as a framework in which we position our work.

Business value viewpoint
We describe services as economic activities, in which customers and suppliers exchange economic values. Hence, we position our work in the business value viewpoint. Traditionally, this viewpoint has been the realm of business researchers, who use natural language for knowledge representation. In recent years, authors as those mentioned in Section 3.2.1 employ structured modeling practices in the business value viewpoint, introducing also new knowledge representation techniques: UML diagrams, ontology editing software tools as *Protégé* and graphic visualizations. Also our research uses structured modeling techniques to develop a service ontology from a business value viewpoint, using knowledge representation techniques as UML diagrams, ontology tools as *Protégé* and *OntoEdit* and graphic visualizations. Using these knowledge representation techniques we allow for a computer-based analysis of business issues, adopting the viewpoint of business analysts and marketeers.

Business process viewpoint
After having used the business value viewpoint to describe *what* is offered *by* whom *to* whom, the business process perspective is used to describe *how* these service offerings are selected, negotiated, contracted and provisioned *operationally*, and may include activities that are performed by humans and/or by information systems. A remark has to be made here about the term 'activity'. This term has been used by us in our value-driven definition of the term 'service', but is also used often to describe business processes. So what is the difference between an 'activity' in the context of the business value viewpoint, and an 'activity' in the context of the business process viewpoint? In the business value viewpoint we consider an activity as an act of economic nature. A service is defined as an economic activity, focusing on *what* is offered *by* whom *to* whom. In the business process viewpoint, on the other hand, we consider the operational nature of an activity, focusing on *how* these service offerings

(being economic activities) are operationalized. For example, seen from the business value viewpoint, an ISP service is an activity in which customers pay a certain amount of money to an Internet Service Provider, and possibly commit themselves to consume this service for a given period of time, in return for an Internet connection with a predefined speed, and a helpdesk support via the telephone. Seen from the business process viewpoint, the same service is a series of operational tasks: customer registration, connecting the customer to a physical network, assigning an IP address to this customer, billing and more. A single business process may be part of the operationalizing of multiple services; this is typically the case for supporting activities as billing and customer registration. An extensive discussion on the differences between modeling activities in these two viewpoints is given in Gordijn et al. (2000).

Various notations have been suggested to model business processes, varying in their degree of formality and intended users. These include UML activity diagrams (Fowler & Scott 1997), Event-Driven Process Chains (EPC) (Keller et al. 1992, Mendling et al. 2005), Integration Definition for Function Modeling (*IDEF0 website* 2005) and Petri nets (van Hee 1994).

Information system viewpoint
Services that are offered online are supported by information systems. The information system viewpoint describes the components of an information system. UML techniques as activity diagrams and interaction diagrams can be used to represent this viewpoint. Online service provisioning always involves an information systems supported process. On the one extreme, only a small fraction of the business process may be performed online (e.g., providing information about an offline service), and on the other extreme, the whole service offering may be performed online, including the search for services, negotiation, contracting, delivery and post-sales customer service. Such service offerings and related business processes are potentially realized by Websites and information systems, using a variety of Web standards, techniques and technologies, for example web services: "loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols" (Stencil Group 2001)). Web service technologies are currently being developed to support application integration among business partners in B2B scenarios.

Computer science / artificial intelligence viewpoint
Services that are offered online are supported by information systems. Software components developed within the information system viewpoint use underlying knowledge bases. These are formal representations of parts of the world, about which information systems have to reason. For example, web services can be supported by a semantic markup to facilitate automation of tasks as service discovery, execution and composition and interoperation (McIlraith et al. 2001). These tasks are supported

Viewpoint	Focus	Representation techniques
Business value	Creating and exchanging economic values	Natural language
		UML-based diagrams, graphic visualizations and ontology tools as Protégé
Business process	Process ownership, scheduling, required resources	UML, EPC, IDEF0, Petri nets
Information system	System components	UML
Computer science / artificial intelligence	Upper level and domain ontologies	DL-based ontologies, OWL

Service Ontology

Figure 3.1: Positioning the service ontology

by OWL-S, a web service ontology which provides a core set of markup language constructs to describe web services unambiguously in a computer-interpretable form (OWL Services Coalition 2004). Main differences between our definition of services (in our service ontology) and between OWL-S web services are (1) the level of focus (economic value of a service vs. service process); (2) the notions of time and scheduling (they do not exist in our work, as we focus on *what* is offered; they are a central element in OWL-S). Knowledge representation techniques in this viewpoint are Description Logic-based ontologies and the semantic web standard ontology language OWL.

Figure 3.1 summarizes the above discussion, and highlights the positioning of this thesis in the spectrum of e-service design among other researchers who use structured modeling techniques within the business value viewpoint.

3.2.3 Introduction to the Service Ontology

In the next sections we present our service ontology, using examples from a credit card service, for which we assume all readers to be familiar with. We present and exemplify the various concepts that constitute the service ontology. In Chapters 4 and 5 we show how we use these constructs to reason with the service ontology.

Our service ontology comprises of two distinct perspectives, describing how two different stakeholder groups view services: customers and suppliers. As argued in Chapter 2, customers are typically not interested in a service itself, but in the value – the benefits – thereof. These benefits provide a solution for customer needs and demands. In view of this, our ontology includes two sub-ontologies, or perspectives, to describe the needs of customers (demand-side perspective) and the services that suppliers provide, in terms of benefits that they provide (supply-side perspective).

The *service value* perspective is a demand-side, customer perspective. It describes the service from a customers' point of view in terms of customers' needs and demands, their quality descriptors and their acceptable sacrifice, in return for obtaining the service (including price, but also intangible expenses such as inconvenience costs and access time).

The *service offering* perspective describes services from a supplier's perspective; it describes service components (service elements), their required inputs and their outcomes, as they are actually delivered by the service provider in order to satisfy customers' needs. Our service description adopts a system theory view (Borst 1997, Borst et al. 1997), such that services are said to have ports and interfaces, similarly to electricity sockets. Service outcomes are often not physical outcomes of a process (as described in process models), but rather the *benefits* that a service provides. This distinction is important to understand, and is due to the following reasons, lengthily discussed before: (1) services often cannot be described in unambiguous terms based on physical characteristics, (2) as argued in Chapter 2, customers are typically interested in the benefits of a service, rather than in the service itself, and (3) a development towards a need-oriented matching between activities of customers and suppliers (Normann 2001) requires suppliers to present their products in such a way that customer benefits are emphasized.

Our service ontology, presented in this thesis, is of a descriptive nature. In creating the ontology that we present in this chapter, as well as in assessing the reasoning capacity we present in the following chapters, we concentrated on implementing business logic. Our model has not been influenced by considerations of computational complexity, which is an important research area by itself. Neither have we implemented a means to select the best solution for a service bundling (or: configuration) task; instead, we present all solutions.

True to our multidisciplinary research environment, we provide two different representations for our service ontology. Graphic visualizations enhance human understanding of models, and serve for communication with non technical stakeholders. A more formal representation, using UML diagrams and a Web based knowledge representation standard (RDFS) provide a degree of formality that enables using the ontology by computer-based systems. To this end, we also provide in Appendix A a list of inherent domain constraints that are not included in the UML and RDFS representations of the ontology, and yet are part of the service ontology.

In following two sections we present the two sub-ontologies: the service value perspective and the service offering perspective. For every concept we discuss, if applicable, its attributes, its relations and its visualization, and we provide an example from a credit card service. We elaborate the most about the service offering perspective, which has been the main focus of our work.

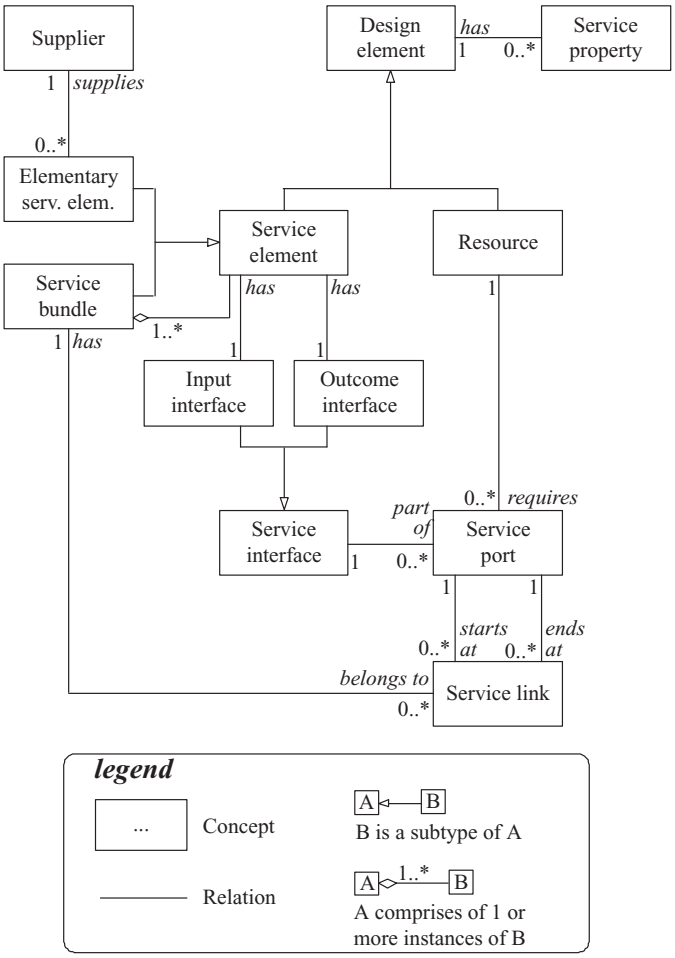

Figure 3.2: Service offering sub-ontology

3.3 The Service Offering Perspective

The sub-ontology representing the service offering perspective is sketched in Figures 3.2, 3.8 and 3.11, using UML class diagrams.

The core ideas behind the service offering perspective – reflecting a supplier view on service offerings – are:

1. A customer is typically not interested in a product (service, good) itself, but in the benefits that the product presents him (Teare 1998, Lancaster 1966).

2. Services represent a value exchange: customer and supplier exchange objects of economic value to which we refer as resources. A service requires some

resources (service inputs) and results in the availability of other resources (service outcomes). Hence resources are not to be understood as operational resources (things needed to carry out a business process); they are typically the benefits of a service, and its costs.

3. A service bundle is in fact a bundle of benefits (Kasper et al. 1999).

4. Suppliers may choose to sell their services only as a package (meaning that services cannot be sold independently) or as an optional bundle (the services can be sold separately or as a package). Other dependencies exist between services (e.g., substitution, exclusion).

5. The bundling process of services is similar to the configuration process of components, where an artifact (bundle) is created, given a set of components (services), parameters and constraints (e.g., dependencies between services), and given a set of requirements (e.g., benefits that a bundle should present). This topic will be discussed in detail in Chapter 4.

In the following sub-sections we present the concepts that constitute the service offering perspective of our ontology. We divide them into four groups, based on the knowledge that they represent:

1. Concepts for modeling services (service bundles or more elementary services) as bundles of benefits, in accordance with business research.

2. Concepts for modeling services (service bundles or more elementary services) as components, in accordance with configuration theory.

3. Concepts for modeling business rules that act as constraints on how services can be configured into service bundles.

4. Concepts for modeling customer requirements for configuration (service bundling) (in fact, concepts for modeling a desired service bundle).

3.3.1 Constructs for Modeling a Service as a Bundle of Benefits

Service element. Service elements are what we defined earlier as "economic activities, deeds and performances of a mostly intangible nature". The service marketing and management literature distinguishes between three types of service elements, from a supplier perspective: a core service (the main business), a supplementary service with a supporting role (making the core service possible) or a supplementary service with an enhancing role (improving the service's value by adding extra features) (Grönroos 2000, Kasper et al. 1999, Lovelock 1983).

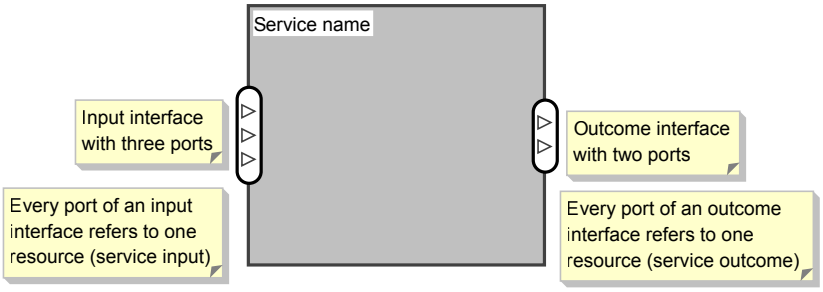

Figure 3.3: Service element

- Core service. A core service describes how the supplier's business adds value to a value chain. This is the reason for the supplier's presence on the market. A firm may have multiple core services, e.g., banking facilities as well as insurances.

- Supplementary service. A service that accompanies the core service/product, ranging from finance to training. It may be of two types:

  - Supporting supplementary services are needed in order to enable the core service consumption. In the absence of these services, it is impossible to consume the core service.
  - Enhancing supplementary services are often considered to be the elements of the service that define it and make it competitive. They increase the value of the service, or differentiate it from those of competitors (Grönroos 2000); the core service can however be consumed without them.

  To avoid confusion, note that the use of the terms supporting and enhancing services is author-dependent. The services that we refer to as supporting, respectively enhancing, are called facilitating services and supporting services respectively by Grönroos (2000).

Our work, on the other hand, is customer oriented, due to the increasing customer-oriented matching between activities of customers and suppliers. Customers are typically not interested in whether or not the service they want to consume is considered as the core business of a supplier or not (they are often not aware of terminology as "core service"). Moreover, customers are often not aware of the existence of supporting services (typically, logistic services). Consequently, we do not use above typology for service elements as such. We will refer to this subject again in Section 3.3.3, when we discuss service dependencies.

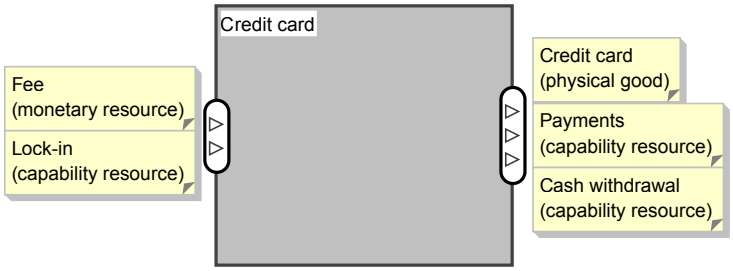

Figure 3.4: Credit card service element

Attributes. A service element has a name with which the a supplier presents the service to its customers.

Relations. A service element has exactly one input interface, and one outcome interface. The concept 'service element' has the relation 'has context', which has been omitted from Figure 3.2. The concept 'context' is described below, in Section 3.5.

Visualization. A service element is depicted as a rectangle with its name on the top left corner, and with two interfaces: an input interface on the left hand side, and an outcome interface on the right hand side (see Figures 3.3 and 3.4).

Example. A credit card service; a travel insurance service (possibly as an enhancing service, combined with a credit card service).

Elementary service element. A service element may be decomposed into smaller service elements, as long as the smaller elements can be offered to customers separately or by different suppliers. Once a smaller element represents a non-separable service element that is offered by one supplier, we call it an elementary service element. The decision whether or not to split a service into smaller services depends on the supplier, as we will show in the example services.

Relations. An elementary service element is supplied by exactly one supplier.

Visualization. An elementary service element is a service element, and therefore adheres to Figure 3.3 as well.

Example. A credit card company may offer the credit card as a payment facility and/or for money withdrawal from ATMs. While one credit card company may consider both functionalities to be non-separable, another may offer credit cards either only as a means of payment, or – for a higher price – also for money withdrawal from ATM's. The first supplier will typically model just one elementary credit card service, while the other will model two elementary services: a payment service and a cash withdrawal service. Together, these two services will offer a similar offering to the credit card service of the first supplier.

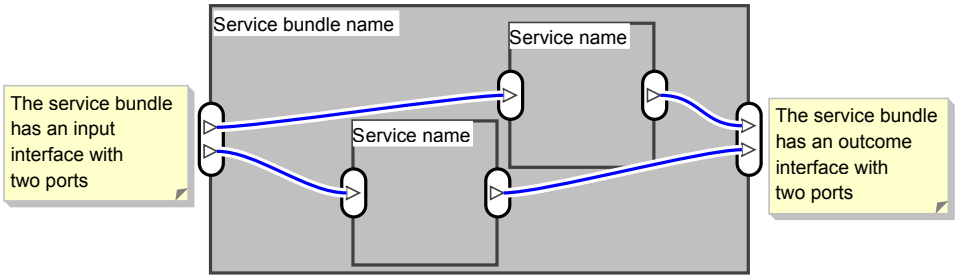

Figure 3.5: Service bundle

Service bundle. A service bundle is a complex service element, including one or more service elements, any of which may be either elementary or a bundle. It has no explicit restriction on the number of suppliers that may participate in this offering (but as mentioned before, every elementary service element – and thus also every elementary service element within a service bundle – is supplied by exactly one supplier).

Relations. A service bundle includes service links (see Figure 3.5); these will be discussed in Section 3.3.2.

Visualization. A service bundle is a service element, and therefore adheres to Figure 3.3 as well. Yet, it introduces extra complexity, since it includes other services as well, as depicted in Figure 3.5. The service elements within a bundle are visualized as service elements within a larger service element (service bundle).

Example. A service bundle may include a credit card service and a travel insurance service. The credit card service may also be a bundle, including a payment service and a cash withdrawal service.

Supplier. An entity, mostly with a legal status, that offers services.

Attributes. A supplier has a name, e.g., a company name or the name of a governmental organization.

Relations. A supplier supplies elementary service elements. Note that as a service bundle may include multiple elementary service elements, it may be supplied by a group of suppliers.

Example. American Express, MasterCard, Visa.

Resource. The provisioning of a service mostly requires some benefits to be sacrificed (service inputs), and results in the availability of other benefits (service out-

comes)[1]. The outcome resource of one service may be used as an input resource of another service. We refer to both as resources. As emphasized before, inputs and outcomes reflect the value exchange between customers and suppliers, and not a process-level flow of resources that are transformed into an end-product. The resources we refer to describe *what* is being offered, and not *how* the service is made operational. We identify the following types of resources:

- *Physical goods*. Sometimes described as 'those things that can be dropped on the floor'. Quite often the result of an interaction between customer and supplier may be that the customer has something of a tangible nature, like airline tickets or a credit card. The prime goal of these objects – referred to as *physical evidence* or *tangible evidence* – is often not the possession of something tangible, but supporting a certain image of a service, so that customers judge the service positively (Shostack 1977). When services are added to a physical good (like car insurance and a car), these services may look more tangible but in fact the service itself remains just as intangible as services that are not added to a physical good (Kasper et al. 1999).

- *Human resources*. Human resources may refer to the supplier (i.e., employees) or to the customer (own participation in the service provisioning). We do not model human resources if they are inherent to the service. For example: if a customer orders a credit card, it is obvious that (1) the customer spends some time on the ordering activity, and that (2) an employee will handle the transaction (or at least those parts of it that cannot be automated). We model human resources where they reflect costs or value for the customer. For example, customers who serve themselves in a self-service restaurant sacrifice some of their comfort and experience (of being served) in return for a lower price.

- *Monetary resources*. Mostly money, but one could also consider stocks or other value-papers. The value of monetary resources is determined by a formula (a pricing model, to be described below), set by the supplier (e.g., usage-based price), and possibly subject to negotiations.

- *Information resources*. Information may be of economic value, for example in a news provisioning service or in a weather report service. Suppliers often value information about their customers, when trying to increase customer loyalty. Since suppliers are willing to reward customers for this information, it has economic value and is a resource. We do not model information flow which is

[1] The notions service input and service outcome refer to the resources in the input interface, respectively outcome interface of a service. However, these terms are not part of the ontology, as their meaning is captured by other terms: *resource*, in combination with *input interface*, and *resource*, in combination with *outcome interface*.

necessary to carry out a business process, such as customer name or account number.

- *Capability resources*. People often appreciate having the *ability* to do things (even if they eventually do not do them). For example, people live in the center of town so that they can easily enjoy all the facilities of the town, but in fact they may use these facilities only once every so often. When buying an insurance we pay for the ability to receive some service in case something goes wrong, but often we eventually do not use that service, because nothing goes wrong. Hence, we pay for a *capability*.

- *Experience resources*. Every service involves a service experience. The experience becomes a resource however, when it reflects costs (a service may have psychological costs, e.g., bad smell) or value (e.g., an added value of going to Euro Disney is having fun; a Gold credit card is a status symbol) for the customer.

- *State-change resource*. Services are "activities... of bringing about a desired change in – or on behalf of – the recipient of the service" (Lovelock 1983). The object of change may be the customers themselves (e.g., haircut, medical treatment), physical goods (e.g., car repair, shipment of goods) or information (e.g., translation services). In some services the change can be related to a property of some resource, whereas in other services the subject of the state-change is not a resource, e.g., a passenger taking a flight undergoes a state change, but he is not a resource. In such cases the economic value of a service, from the customer point of view, is a change of state: the customer was in Amsterdam, and now he is in Sydney. He pays for this change of state.

Attributes. A resource has a name, a type and three more attributes that are of importance for the software-aided bundling process:

- *Consumability*. A resource is said to be *not consumable* if it is still available for consumption, after having been consumed already. When bundling service elements A, B and C (and possibly more) into service bundle X, service element A may have a service outcome that is required as a service input for service elements B as well as C (and possibly more). If this resource (outcome/input) is not consumable, service A can provide this resource as an input for both B and C, and not only one of them. In such a case, this service outcome of service element A will also be part of the outcome interface of service bundle X (implying that it can be consumed by an external entity). The concept consumability is depicted in Figure 3.6.

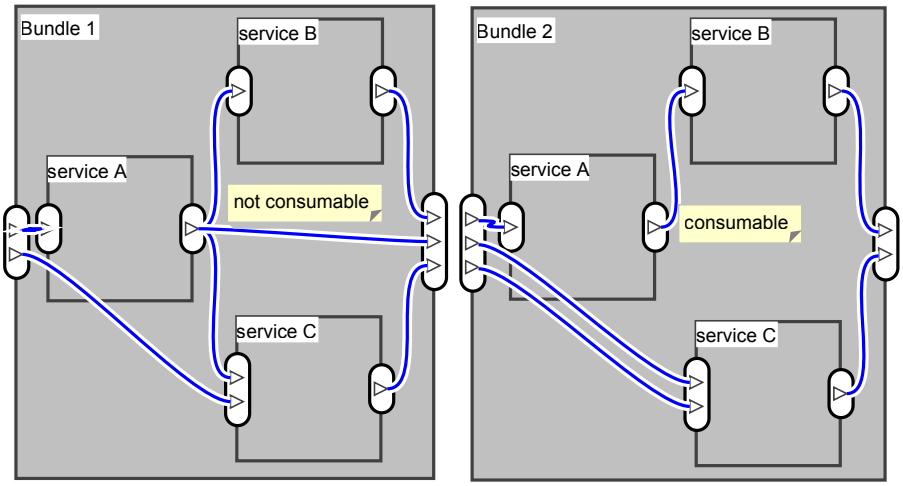

Figure 3.6: Consumability. Note that the outcome resource of service A in bundle 1 is not consumable, hence it can be consumed a number of times, while the outcome resource of service A in bundle 2 is consumable, hence it can be consumed once only.

- *Compositeness*. Refers to whether two or more resources of the same type can be united and modeled as one resource, when they appear in the same interface. Some resources may not be united into one resource, for example tables and chairs – both pieces of furniture (physical goods) – normally cannot be modeled as one resource. On the other hand, a common example for the compositeness property is the (financial) costs of services. If two service elements within a service bundle require a fee, we do not necessarily model two different fee inputs in the input interface of the bundle; instead, they may be united to one fee input. Likewise, if an input of 1000 Euro is available for a bundle, and the bundle includes two services that require 500 Euro each, the input of 1000 Euro may be decomposed into two inputs of 500 Euro.

- *Has Consumable Property*. If a resource is consumable, it has at least one service property that "gets consumed", meaning that the value thereof decreases upon consumption. This is referred to as 'consumable property'.

Relations. A resource may be related to multiple service ports.

Visualization. A resource can be marked by its name nearby a service port (to be discussed below), or on a service link between two ports (see below). Examples are shown in Figure 3.4.

Example. Several resources are depicted in Figure 3.4: one monetary resource, one physical good and three capability resources:

1. Lock-in: customers commit themselves to consume this service for a given period of time (typically one or two years); this commitment is valuable for the supplier.
2. The ability to pay worldwide.
3. The ability to withdraw money from ATMs.

Design element. This term is borrowed from configuration theory (Schreiber et al. 2000), where configuration is described as a design task. In our case, a design element is the supertype of 'service element' and 'resource', the two main elements that play a role in the service bundle design task.

Relations. A design element has zero or more properties and conditional outputs (to be discussed below).

Example. See examples of service elements and resources.

Service property. Properties encapsulate domain knowledge that describes design elements (resources and services) qualitatively and quantitatively.

Attributes. A property has the following attributes: (1) a name; (2) a value; (3) a value type (numeric, String); (4) a unit to measure the value; (5) an optional description of the property, using natural language; (6) comparability (the latter is used for comparing resources; the comparison process takes into consideration only the comparable resource properties). We compare resources by comparing all the properties of a resource that are marked as comparable. If the property has a numeric value, we compare this value (equal, smaller, larger). If the property has a String value, we compare the Strings (here only the relation equal is possible). A formal definition of equal resources is given in Appendix A.

Relations. The value of a service property may be restricted by a requirement expression (see below), when defining criteria for service bundling.

Example. Property name: fee. Property value: 40. Value type: numeric. Property unit: euro/month. Property description: monthly fee. Comparability: TRUE.

3.3.2 Constructs for Modeling a Service as a Component

Service interface. Every service element – whether it is elementary or a bundle – has exactly two service interfaces: one input interface, and one outcome interface (both are subtypes of the concept service interface). Together they provide a black-box view of a service, abstracting away from its internalities. A service interface consists

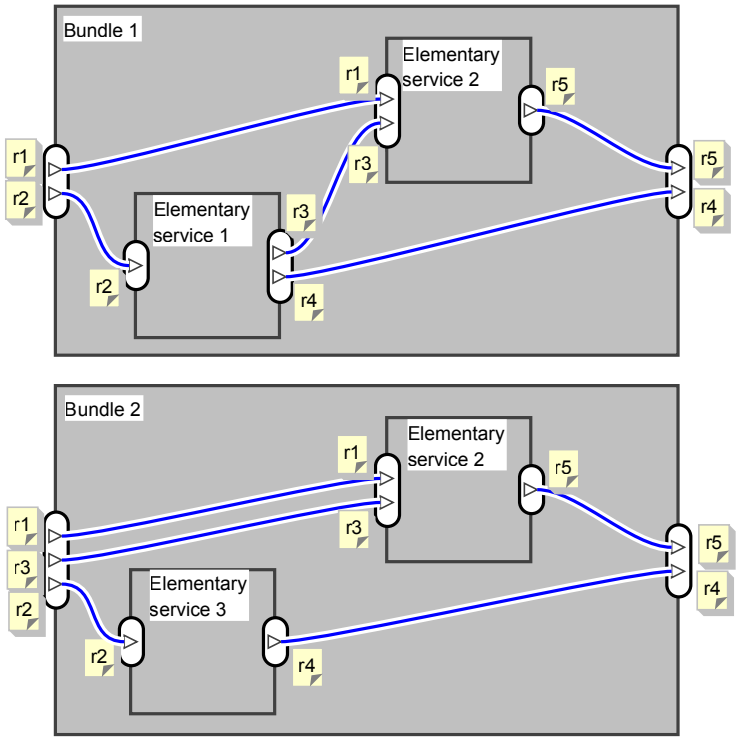

Figure 3.7: Example service bundles. Note how in the bundle 1 resource r3 is satisfied internally: it is an outcome of service element 1, and it is consumed by service element 2 within the bundle. In bundle 2, however, service element 3 does not provide resource r3; consequently, r3 has to appear in the input interface of the bundle.

of service ports (to be described below). Grouping ports into one interface is related to the idea of bundling: either all ports are available, or none at all.

Relations. A service interface is part of exactly one service element; it consists of zero or more service ports.

Visualization. Service interfaces are visualized as rounded boxes at the right and left edges of a service element. Service ports are drawn inside service interfaces (see Figure 3.3 through 3.7).

Input interface. The provisioning of a service requires the availability of certain resources. This is modeled by an input interface: a set of ports (to be discussed below) and associated resources must be available to make possible the provisioning of a service. The input interface does not specify who has to provide the required resources (e.g., a customer, or another service), how the service is provisioned or what are the outcomes of the service.

When customers buy multiple services as a service bundle, all the input interfaces of these separate services must be satisfied. A subset thereof may be satisfied internally, meaning that the outcome interface of one service may provide resources that can satisfy some of the ports of an input interface of another service within the bundle. All (parts of) the input interfaces of all services within a bundle that are *not* satisfied internally in the bundle, appear in the input interface of a bundle, meaning that they all have to be provided by an external entity that wishes to consume the service bundle. Thus, an input interface of a service bundle is the union of the input interfaces of the service elements inside the bundle, minus the ports that are satisfied internally. This is depicted in Figure 3.7.

Visualization. An input interface is depicted as a service interface (rounded box) at the left edge of a service element.

Example. Figure 3.4 shows an example service element, where the input interface consists of two ports. Both of them must be satisfied (i.e., their associated resources are provided), for the service to be provisioned. However, often inputs are provided only *after* the service has been provided.

Outcome interface. The provisioning of a service results in the availability of certain outcomes, referred to as "a bundle of benefits" (Kasper et al. 1999). This is modeled by an outcome interface: a set of resources become available once a service has been provisioned. The outcome interface models the idea of bundling: the set of outcomes (benefits) in an outcome interface is non-separable in the service to which a given outcome interface is attached. Once again, the interface abstracts away from how the service is provisioned, or which input resources are required for the provisioning of the related service. Similarly to the description we gave for input interfaces, also an outcome interface of a service bundle is the union of the outcome interfaces of the service elements inside the bundle, minus outcome ports that provide resources that are consumed internally.

Visualization. An outcome interface is depicted as a service interface (rounded box) at the right edge of a service element.

Example. Figure 3.4 shows an example service element, where the outcome interface consists of three ports. They represent the "bundle of benefits" that the given service presents to customers.

Service port. A service port indicates a certain resource that is either a pre-requisite for carrying out this service element (input port), or that is the result (outcome) of carrying out this service element. A service element is then characterized by its required inputs and by the outcomes it produces. The notion of ports stems from the technical system theory (Borst 1997). Ports create an interface to which external elements must adhere, if they wish to interact with a service, abstracting away from the

internal structure or operationalization of a service.

Attributes. We discussed earlier the compositeness attribute of resources. Also a port has the attribute *is composable*. When associated with a resource, this attribute describes whether the resource is of a composable nature. However, sometimes a business may decide not to compose resources even though they are of a composable nature. For example, two fee resources of two elementary services in a service bundle may be composed into one fee (implying one bill for two services) or not (implying that every service generates its own bill). Although the nature of money resources is composable, businesses may decide not to compose these resources (and to send two separate bills). Consequently, also a port has the compositeness attribute, enabling to define per case whether two composable resources should indeed be composed. Hence, resource compositeness describes whether a resource is of composable nature, while port compositeness describes whether the resource in a specific port should be composed (assuming that the resource is also of a composable nature, and that two or more similar resources exist in one interface). For a similar reason a port also has the attribute *is consumable*.

Relations. A service port requires exactly one resource. It belongs to exactly one service interface, and may be connected to service links (to be discussed below).

Visualization. The service port is depicted by a small triangle (see Figure 3.4). The name of the resource related to the port can be depicted with a text label.

Service link. A service link is a connection between two ports *of different service elements* in a bundle. It should be interpreted as "the port where the link starts provides a resource for the port where the link ends". This has the following implications: (1) a service link has a direction; (2) a service link exists only in a service bundle, because it always connects two interfaces of two services (either two services inside a bundle, or the bundle itself with a service within the bundle); (3) no service link is allowed between two ports of the same service, because a service cannot provide a resource for itself.

Generally speaking, two ports may be connected by a service link if their resources are identical (and subject to other conditions). However, they need not always be identical; if an input of 1000 Euro is required, and an input of 2000 Euro is available, we should be able to use the available 2000 Euro even though we need less. This is facilitated by the comparability attribute of resources, described earlier in this section.

A service element (say, service X) may be part of multiple bundles. In every such bundle the same ports of service X will be connected differently to other ports by service links. See for example Figure 3.7, where elementary service 2 has a port that requires resource r3. This port is connected to different ports in bundle 1 and bundle 2. As a result, we need a mechanism to relate a service link to a bundle, so that we can tell which of the links of a given port to use in a given case. A service link is

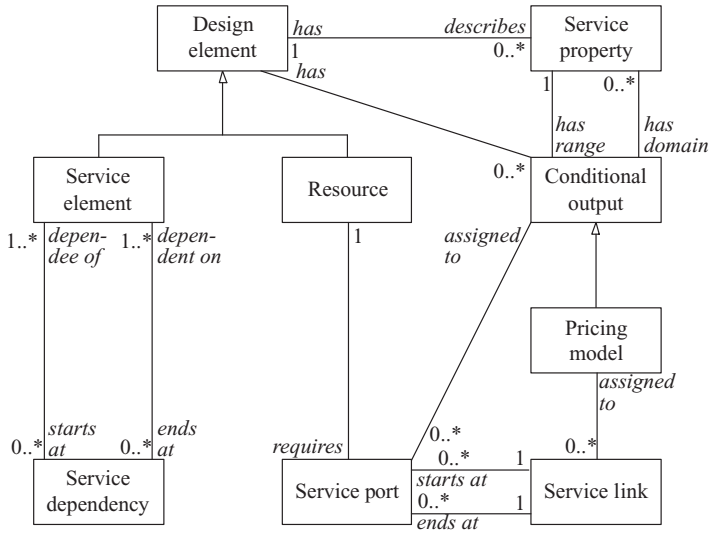

Figure 3.8: Business rules/constraints: service dependencies and conditional outputs

therefore not only a relation between two ports, but also a relation between a service element (elementary or bundle, say service X) and a service bundle (say bundle Y), that includes service X.

Relations. A service link starts at exactly one service port, terminates at exactly one service port, and belongs to exactly one service bundle.

Visualization. A service link is shown as a line between service ports. The name of the resource which is being provided may optionally be presented by a text label nearby the link.

3.3.3 Constructs for Modeling Business Rules

The service offering perspective includes constructs to define two types of constraints (Gruber et al. 1996), descriptions that limit the permissible values of characteristics of service elements and represent supply-side business logic. These are referred to as conditional outputs and service dependencies (see Figure 3.8). We will discuss the notion of constraints also in Chapter 4, where we focus on how services are *configured* into service bundles.

Conditional output. Constraints on the possible values of properties of design elements are referred to as "conditional outputs". We call them conditional outputs because they include a condition, which determines the value of some property (the so-called 'output'). They therefore have the form "IF...(condition) THEN a prop-

erty's value is... (output)". We distinguish between three types of conditional outputs.

1. A conditional output refers to one design element (e.g., the value of the property "location" of the resource "cash withdrawal" may be either "NL" or "worldwide").

2. A conditional output refers to two or more resources within one service elements (e.g., formula that determines the price based on the quality level of an outcome).

3. A conditional output refers to two or more resources in two or more services (e.g., if the fee resource in service X has value 1000, then the invoice resource in service Y has value 1000 as well).

In all three types the THEN part of a conditional output refers to the value of exactly one property; they differ however in the IF part. In the first type, the IF part of the condition does not refer to any property. This can be expressed as "IF (TRUE) THEN property N = ...". In the second and third types the IF part of the condition refers to one or more properties of some design element(s). This can be expressed as "IF (property X == ... AND property Y == ... AND...) THEN property N = ...".

Attributes. A formula of the type IF...THEN describes the conditional output.

Relations. We refer to the IF part of a conditional output as 'domain', and to the THEN part as 'range'. A conditional output (see Figure 3.8) determines the value of exactly one property (the 'range') of a design element. This is modeled with the relation "conditional output hasRange... service property (cardinality 1)". The value of that property may be dependent on some other property or properties (the second and third types of conditional outputs), or on no property (the first type of conditional outputs). These properties are the conditional output's 'domain'. We model this dependency with the relation "conditional output hasDomain... service property" with cardinality zero or more. Some resources allow a range of values, and are assigned to multiple services. Their properties values are determined in every such service element by conditional outputs. Hence we assign conditional outputs to service ports in order to determine the value of a resource assigned to this service port, when the same resource may behave differently in every service element to which it is attached.

Pricing model. Pricing models are a special type of conditional outputs. A pricing model is a representation of how a company plans to set its prices (Daly 2002). In other words, a pricing model is a representation of how a price is derived. It can be expressed in natural language, or as a mathematical formula. Examples of broadly used pricing models are:

- *Flat-rate*: the user pays a fixed amount which is independent of usage (Choi et al. 1997).
- *Usage-based*: the user is charged on basis of usage (Essegaier et al. 2002).
- *Two-part tariff*: the user pays a fixed amount plus an additional usage-based charge (Essegaier et al. 2002, Dolan & Simon 1996).
- *n-block tariff* (where n = 2): a price per unit is charged up to a certain quantity (say: X), and then a different price per unit is charged for all units greater than X. This pricing model is applicable when each block has both a fixed and variable price component. It can be generalized for n $>$2 (Dolan & Simon 1996).

By adding pricing models to a service ontology we allow suppliers to determine the price of a service (elementary or bundle). That price plays a significant role in determining the value for the customer, and also gives a perception of quality (Payne 1993). An extensive report on pricing models in the service ontology can be found in de Miranda (2005).

Relations. The pricing model concept is a subtype of the conditional output concept. It inherits the domain/range relations of 'conditional outputs', using which it determines the value of one property – price – based on values of other properties. A pricing model has relations with the concepts 'service port' and 'service link':

- A pricing model is assigned to... service port (cardinality: 0..*). This relation is inherited from the supertype conditional output. Restriction: a pricing model can be attached to a service port that requires a monetary resource only.
- A pricing model is assigned to... service link (cardinality: 0..*). Some services may use a specific pricing model only when they are part of a specific bundle. A service link is a connection between two ports (in a bundle), and is also a relation between a service element (elementary or bundle, say service X) and exactly one service bundle (say bundle Y), that includes that first service element. By adding a pricing model to a service link, we model that this pricing model is used only when service X is part of bundle Y. The service link then connects two monetary resources: that of service X and that of bundle Y. Consequently, it becomes possible to calculate the price of bundle Y based on the pricing model that is in the link between ports of service X and bundle Y (see Figure 3.9). Note: as can be seen in Figure 3.9, a service link may also connect ports of two services X and Z that are both included in service bundle Y. In such a case, if a pricing model is attached to this link, it will have no influence on the ports of service bundle Y, and will therefore not be used directly to determine the price of service bundle Y.

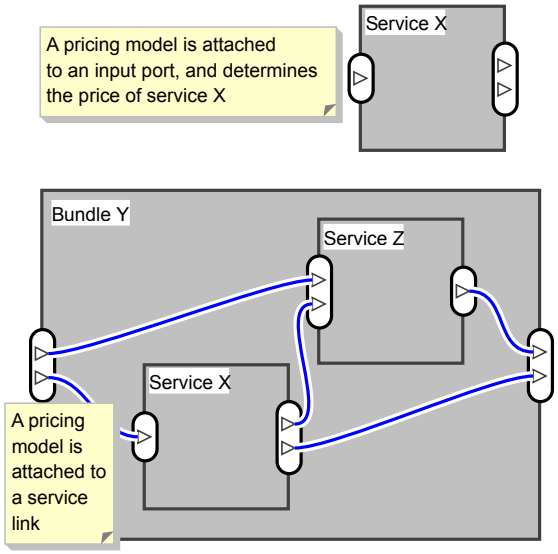

Figure 3.9: Pricing models. In the top image, a pricing model is attached to the input port of service X, to determine the price of the service. Once service X is included in service bundle Y (lower image), a different pricing model is attached to a service link between ports of service X and bundle Y. This latter pricing model determines the price of service X, when service X is included in service bundle Y, and overrides the pricing model that was attached to the input port of service X.

Example. Credit card suppliers often use a flat rate: the user pays a fixed amount per year. Another possibility is a two-part tariff: a fixed amount plus a fee per transaction.

Service dependency. Service dependency is a relationship between two or more service elements that defines a dependency between these service elements. It represents a constraint on how these service elements may or may not be bundled. Unlike conditional outputs that refer to the internal characteristics of a service element, a service dependency is a constraint on the configuration (or: bundling) of services, rather than on the service elements themselves. A service dependency is defined as a formula, that receives two inputs of type 'service elements' (A – the *dependees* – and B – the *dependents* – are *disjoint* sets of service elements), and produces as output a set of possible configurations of these two inputs. In most cases A and B comprise of exactly one service element, but in developing real-world studies we encountered situations where the cardinality was more than one. When the sets of dependees or of dependents include more than one service element, for example the set of services {service 1, service 2,... service n}, we interpret this as {service 1 OR service 2 OR ... OR service n}. Service dependencies are driven by business logic. For example:

- Two services provide the same benefits for customers, and can therefore be used as substitutes.
- Two service elements that use a same business process can be bundled to reduce operational costs.
- A travel insurance service adds value to a credit card service. It is a common marketing practice to add such value-adding services to a core service in order to differentiate the core service and attract customers. From a customers' perspective, the credit card will still be their core service, the main service they want to buy. The extra ('free') travel insurance may convince them to choose for a specific credit card, rather than for another credit card service that does not offer this enhancing service.

Every service element may have a number of service dependencies, but there may not exist more than one service dependency between the same two services. We define the following types of service dependencies, with A, B as two disjoint sets of service elements. These are described below, assuming that A includes several services x, y,... z , and B includes several services a, b,... n (yet, in practice in most cases A and B include only one service element). Note that service dependencies are not symmetric, so it is important whether a service is the first argument or the second argument of a service dependency.

For every pair of service elements $x \in A$ and $n \in B$:

1. **Core/enhancing (*A, B*):** n is an enhancing service of x (and thus x is a core service of n). Hence service x is a main service a customer is interested in, whereas service n: (1) is not required for the provisioning of service x; and (2) adds value to x; and (3) is an optional service element, next to x; and (4) is not sold independently of service x. If customers wish to consume service element x, they are presented with the option to consume also n, but they are not obliged to consume n.
   Notation: CE(A, B)
   Output: $\{A\}$,$\{A,B\}$ in case both sets include just one service element; otherwise
   $\{x\}$,$\{y\}$,... $\{z\}$,$\{x,a\}$,$\{x,b\}$,... $\{x,n\}$,$\{y,a\}$,$\{y,b\}$,... $\{y,n\}$,$\{z,a\}$,$\{z,b\}$,... $\{z,n\}$

2. **Core/supporting (*A, B*):** n is a supporting service of x (and thus x is the core service of n). In business logic terms it means that x cannot be provisioned without n and that n is not offered independently of service x. Very often n will not present value as such for customers (e.g., billing services), but yet it must be provided to enable the provisioning of x. If customers wish to buy service

x, they are obliged to consume service *n* as well.
Notation: CS(A, B)
Output: {*A,B*} in case both sets include just one service element; otherwise
{*x,a*},{*x,b*},...{*x,n*},{*y,a*},{*y,b*},...{*y,n*},{*z,a*},{*z,b*},...{*z,n*}

3. **Bundled (*A, B*):** If customers wish to buy service element *x*, they are obliged to buy also *n*. This is similar to the *core/supporting* dependency. However, in case of a *bundled* dependency, *n* may be offered independently (remember that this is not a symmetric relation), and the reason for the obligatory consumption of *n* is not that *n* is required to support the provisioning of *x*. In the case of *bundled* services, the bundling is required due to some business logic, such as cost efficiency reasons, marketing reasons or legislation.
Notation: BU(A, B)
Output: {*A,B*} in case both sets include just one service element; otherwise
{*x,a*},{*x,b*},...{*x,n*},{*y,a*},{*y,b*},...{*y,n*},{*z,a*},{*z,b*},...{*z,n*}

4. **OptionalBundle (*A, B*):** Two services *x* and *n* are offered separately, but also as an optional combination (bundle). This is referred to as *mixed bundling* in the literature (Guiltinan 1987, Normann 2001, Barrutia Legarreta & Echebarria Miguel 2004). In most cases, the bundling of two such service elements presents added value to the supplier (e.g., lower operational costs) as well as to the customer (lower price). Unlike the *core/enhancing* service dependency, in the *optional bundle* case service *n* can also be offered independently of service *x*.
Notation: OB(A, B)
Output: {*A*},{*A,B*} in case both sets include just one service element; otherwise
{*x*},{*y*},...{*z*},{*x,a*},{*x,b*},...{*x,n*},{*y,a*},{*y,b*},...{*y,n*},{*z,a*},{*z,b*},...{*z,n*}

5. **Substitute (*A, B*):** Customers consider service *n* to satisfy them at least as much as service *x* (and possibly better). In such cases it is very likely that the service outcomes of service *x* are also made available by service *n* (but *n* possibly offers more benefits). Service *n* can therefore be bought instead of *x*; customers can choose which one of them they prefer.[2]
Notation: SU(A, B)
Output: {*A*},{*B*} in case both sets include just one service element.

[2] As we explain in Chapter 4, the *substitute* service dependency is not required for service configuration, as it represents redundant information. We nevertheless model it, because substitution is an important notion in business research and practice, and because it is useful for designing software tools for service configuration.

There is no logic in including more than one service element in A. If B includes multiple service elements, the output of this service dependency is $\{x\},\{a\},\{b\},\ldots\{n\}$

6. **Excluding (*A, B*):** If service element x is consumed, service element n may not be consumed, for example because x and n are competing services, and suppliers do not want to provide them together, or because legislation prohibits selling both services together.
   Notation: EX(A, B)
   Output: $\{A\}$ in case both sets include just one service element. It makes no sense to model this dependency with A and B that include multiple services.

Attributes. A service dependency has a name (e.g., Excluding, Substitute) and three attributes that describe its behavior (output): whether the first argument alone is a possible output of the dependency (that is, whether $\{A\}$ is a possible solution), whether the second argument alone is a possible output (that is, whether $\{B\}$ is a possible solution), and whether the combination of both arguments is a possible output (that is, whether $\{A,B\}$ is a possible solution).

Relations. The first argument of a service dependency is described by the relation 'service dependency starts at... service element' (cardinality one or more); the second argument of a service dependency is described by the relation 'service dependency ends at... service element' (cardinality one or more).

Visualization. Most service dependencies are relations between exactly two service elements. These are visualized by a bi-directional arrow (see Figure 3.10) with labels that stand for the name of the dependency. Since an arrow is bi-directional, it consists of two service dependencies: DEPENDENCY(A, B) and DEPENDENCY(B, A). In an arrow between two services A and B, the service which is closer to the label is argument B. For example, in Figure 3.10 we see three service dependencies: SU(Service 1, Service 2), BU(Service 2, Service 3) and EX(Service 1, Service 3). This is interpreted as (1) service 1 can be substituted by service 2 (but not vice versa); (2) whenever service 2 is consumed, service 3 must be consumed as well; and (3) whenever service 1 exists in a service bundle, service 3 cannot be added to the same bundle.

Example. SU(American Express Blue, American Express Gold)

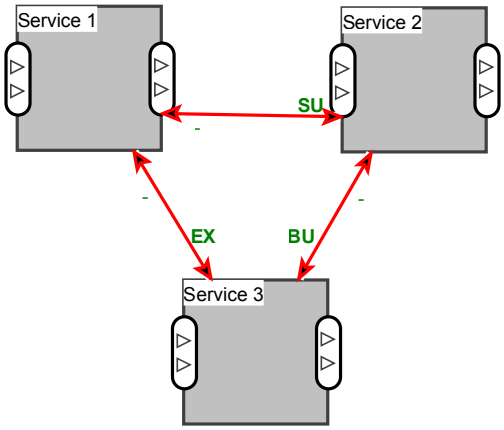

Figure 3.10: Service dependencies: SU stands for *Substitute*; BU stands for *Bundled*; EX stands for *Excluding*; the label '-' is used to denote that there is not service dependency between two services

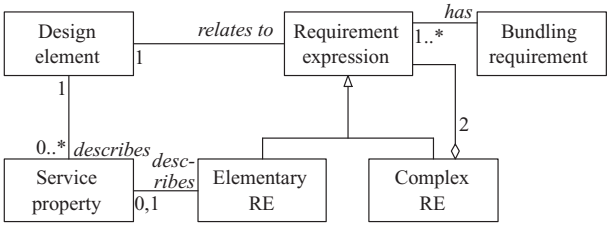

Figure 3.11: Defining requirements for a service bundle in supplier terminology

3.3.4 Constructs for Modeling a Desired Service Bundle

Next, the service offering perspective consists of constructs for defining bundling requirements, specified in terms of requested benefits (resources), as can be seen in Figure 3.11.

Bundling requirement. Requirements for the service configuration process, described as a set of resources (inputs and outcomes) and possibly constraints on their values, are captured by a bundling requirement. In fact, a bundling requirement can be seen as a simulation of a desired service bundle, because it specifies the desired benefits that a bundle should provide. Once a bundling requirement is defined, the goal of the service configuration process is to design service bundles that provide these benefits.

Relations. A bundling requirement comprises of one or more requirement expressions.

Requirement expression. A requirement expression captures all user requirements that are related to one design element that should be part of a desired service bundle.

Attributes. A requirement expression has two Boolean attributes, indicating whether it describes a requirement on an input or a requirement on an outcome. In case the requirement expression is related to a service element, rather than to a resource (both are subtypes of design element), both these attributes will have the value *false*. Otherwise, one of them is *true*, and the other is *false*.

Relations. A requirement expression is related to one design element, requiring its existence in a service bundle.

Elementary requirement expression. An elementary requirement expression may include restrictions on the values that a design element has. If no restrictions are set, it only requires the existence of a specific design element in a service bundle.

Relations. An elementary requirement expression restricts the value of zero or one service property; hence the relation 'elementary requirement expression restricts property'. As explained before, a service property is described by a value, a unit and other attributes.

Attributes. The restriction on a value of a property is defined by the attribute 'elementary requirement expression has a comparison operator': EQUAL, MINIMUM or MAXIMUM.

Example. An elementary requirement expression can require the outcome resource 'Cash withdrawal' (see Figure 3.4) where the property 'location' has the value 'worldwide' with the comparison operator EQUAL. Subsequently, any service that provides the benefit 'Cash withdrawal' but not worldwide, will not be selected as a solution. Similarly, an elementary requirement expression can require the input resource 'Fee' (see Figure 3.4) where the property 'amount' has the value '50', the unit 'euro' and the comparison operator MAXIMUM.

Complex requirement expression. When more than one requirement applies to the same design element, each such requirement is defined as an elementary requirement expression, and they are all grouped into a complex requirement expression, using the logical operators (AND, OR, XOR, NOT) to relate elementary requirement expressions. This feature enables us to define requirements on multiple properties of a design element, or various requirements on the same property.

Relations. A complex requirement expression consists of two requirement expressions, each of which may be elementary or complex.

Attributes. A complex requirement expression has a logical operator (AND, OR, XOR, NOT) which defines the relation between its requirement expressions.

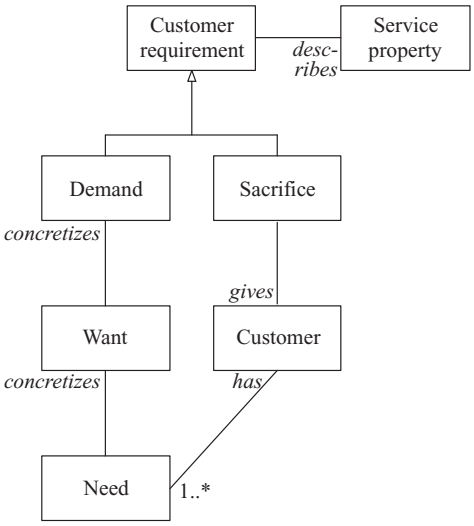

Figure 3.12: Service value sub-ontology

Example. To specify a range as the acceptable price of a service, we use a complex requirement expression, referring to the input resource 'Fee' (see Figure 3.4). It consists of two elementary requirement expressions with the logical operator AND, meaning that both elementary requirement expressions must be met. The first elementary requirement expression specifies that the property 'amount' has the value '50', the unit 'euro' and the comparison operator MAXIMUM. The second elementary requirement expression specifies that the property 'amount' has the value '30', the unit 'euro' and the comparison operator MINIMUM.

3.4 The Service Value Perspective

The sub-ontology representing the service value perspective is sketched in Figure 3.12. It describes a customer viewpoint on services, using what Kotler (1988) identifies as the starting point for the discipline of marketing: the human needs and wants. The term *need* refers to what humans need and want (to buy) (Kotler 1988). Customers have needs, that are being satisfied by products: services and goods. This corresponds with the two perspectives of our service ontology: the service value (customer) perspective describes what customers are looking for, and the service offering (supplier) perspective describes what suppliers offer. Two distinct perspectives are required because customers typically use a different terminology and have a different view on their needs than suppliers (Vasarhelyi & Greenstein 2003).

Need. A human need is a "state of felt deprivation of some basic satisfaction" (Kotler

1988). Needs are often vague; the need for "financial security", for example, can be interpreted in many ways. One customer may interpret it as "have a stable job"; another customer may interpret it as "have some money for a rainy day"; and a third customer may interpret it as "always be able to pay for anything I buy, without taking the risks involved with carrying cash with me". Needs are therefore too abstract to understand what is the actual service (or good) that a customer seeks. Customers therefore concretize their needs by translating them into wants and demands.

Attributes. A need is described by the attribute 'has expression', which defines a natural language description of the need.

Example. The need to 'feel safe'.

Want. Wants are desires for *specific satisfiers* of deeper needs (Kotler 1988).

Attributes. A want is described by the attribute 'has expression', which defines a natural language description of the want.

Example. Worldwide payment facilities.

Demand. Demands are wants for specific products that are backed up by an ability and willingness to buy them (Kotler 1988). As explained before, the service value perspective uses a customer terminology to describe what customers seek. Nevertheless, customers sometimes describe demands also using supplier terminology. This happens when customers are familiar with available services that can satisfy their needs. To conclude, demands – being wants for *specific* products – may be described by customers either in their own terminology, or in supplier terminology. In the latter case, the customer has in fact already defined the desired solution for his need.

Attributes. A demand is described by the attribute 'has expression', which defines a natural language description of the demand.

Example. Credit card.

Sacrifice. Customers have needs, for which they seek satisfiers, in the form of products: services and goods. Based on the principle of economic reciprocity, customers have to accept a sacrifice, in return for having their needs satisfied. Sacrifices represent the costs – both financial and non-financial – that the customer associates with (and is willing to bear when) buying a service. In the short term, the sacrifice is the price of the service. The long-term sacrifice includes not only the price but also relationship costs. Grönroos (2000) distinguishes between three types of relationship costs:

- Direct relationship costs: investment in office space, additional equipment etc.

- Indirect relationship costs: related to the amount of time and resources that the customer has to devote to maintaining the relationship.
- Psychological costs: caused when the staff of a firm feel that they cannot trust a supplier or service provider; unpleasant sensory experience, such as noises and smells. Psychological costs have a major impact on customers' intention to use the services of the same supplier again in the future (Carmon et al. 1995).

The sacrifice a customer is willing to bear must match the quality level of the requested service; otherwise no service can be offered.

Example. In most cases customers measure their acceptable sacrifice in terms of money, considering a maximum amount of money that they are willing to pay for a satisfier for their need. Sometimes also the time (indirect relationship costs) is important, so a customer specifies a maximum period. For example, whereas one customer will be willing to wait three days to have his car fixed, another will not accept a service that takes longer than one day. Other examples are distance traveled, or activities like product assembly or communications with insurers, governmental agencies and third parties to complete the service (Pitta & Laric 2004).

Customer requirement. Demands and sacrifices are subtypes of *customer requirement*.

Relations. A customer requirement may be described qualitatively and quantitatively by service properties.

Example. See examples of the concepts *demand* and *sacrifice*.

Service property. The concept *service property* is used also in the service offering perspective. In the service value perspective it encapsulates qualitative and quantitative knowledge for describing demands and sacrifices:

- Very often this knowledge defines service quality criteria for the demand at hand.
- Other properties are possible, for example the time or location where a service is available. Time and location are often seen as process-level service characteristics, but this is not the case here. We describe time and location only when these characteristics present benefit to a customer. For example, when customers are interested in a regular telephone line, they will not consider the location to be a benefit, because they assume the service is available everywhere. But when customers are interested in a means to withdraw money from ATMs, the location where the service is available (e.g., location: worldwide) is part of the customers' demand, because they know that whereas this characteristic is valuable for them, not all services provide it.

Hence demands and sacrifices are described by service properties, of which a subset is service quality criteria. Service quality is the degree and direction of the discrepancy between a customer's expectations and the perception of the service (Bigné et al. 1997). Customer expectations embrace several elements, including desired service, predicted service and a zone of tolerance that falls between the desired and adequate service levels (Berry & Parasuraman 1991). Expectations are based on word of mouth communications, personal needs, past experiences and external communications from service providers (Zeithaml et al. 1990). At least two widely accepted methods for defining service quality exist: that of the Nordic school (Grönroos 2000) and that of the North American school (SERVQUAL, see (Zeithaml et al. 1990)). Both methods use generic models; using the methods requires domain- and market-specific knowledge on quality definition.

Example. A demand 'credit card' with the properties 'location: worldwide' and 'esteem: high'.

3.5 Relating Perspectives

The service offering perspective describes service elements including their input- and outcome-resources, as well as bundling requirements in supplier-oriented terms. The motivation for doing so is that the service offering perspective aims at configuring the various (e-)service elements of different suppliers in a more comprehensive bundle, and for doing so we need the actual service elements that can be provisioned by these suppliers.

Customers however do not articulate their needs in terms of supplier-oriented requirements but employ their own, subjective terminology for expressing demands. To deal with these demands, we extended our ontology with needs, wants, demands, and miscellaneous constructs: the service value perspective. To come to a customer-need-driven service bundle, knowledge captured by service value constructs must be transformed into knowledge captured by service offering constructs. In brief, customers state their requirements, which can (partly) be satisfied by a series of resources, which in turn are provisioned by service elements (see Figure 3.13). Customer terminology (demands and acceptable sacrifices) is first transformed to resources, which are customer benefits in supplier terminology. Resources are service descriptors; accordingly, services that provide the requested resources will satisfy a customer's demand, and they therefore function as an initial solution (service bundle) to offer to a customer. Similarly, the acceptable sacrifice implies a restriction on the inputs of a service bundle. The final service bundles are generated by applying business rules on this initial bundle. This service bundling – or configuration – process, which we term *serviguration* , is described in detail in Chapters 4 and 5.

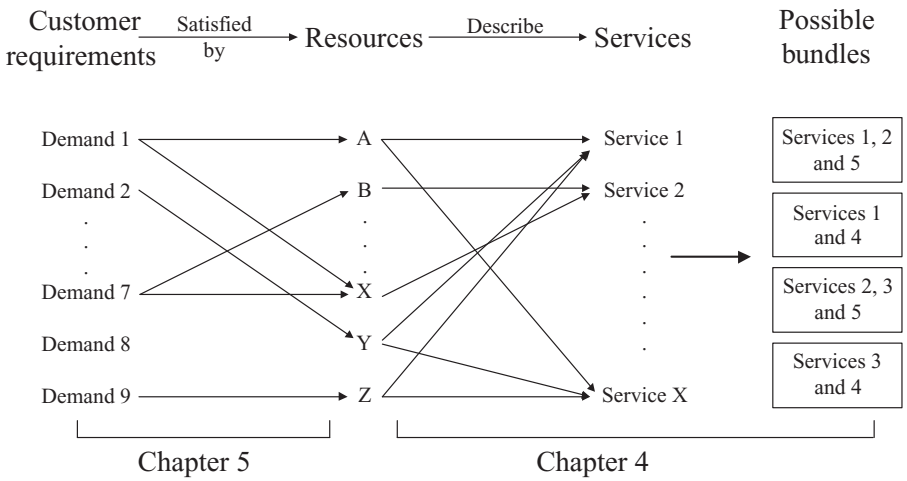

Figure 3.13: Serviguration: configuring service bundles based on customer demands

Serviguration spans over both perspectives: service value and service offering. It is the process of defining bundles of service elements (a supply-side description of services, part of the service offering perspective), that satisfy the customer description of his desired service (service value perspective). The process can be split into two sub-processes:

1. Transformation process between the customer description of the requested service (service value perspective), and the supplier terminology for describing the service. Automating this sub-process requires that (1) customer needs, wants and demands (service value perspective) and available services (service offering perspective) are modeled and expressed in a machine-processable way, and that (2) needs, wants, demands and sacrifices are mapped onto concepts of the service offering perspective. The latter will be the input for the second sub-process – the actual configuration process – resulting in service bundles.

2. Defining zero or more bundles of service elements (service offering perspective) that satisfy this supplier description of the requested service, and thus also the customer description of his requested service.

While the above discussion focuses on customer *demands*, customers may set requirements also on their acceptable sacrifices. These are requirements on the *input* resources of a service bundle. Sacrifices are handled similarly to demands.

In the next chapter (Chapter 4) we discuss the second sub-process of the *serviguration* process: service configuration. In Chapter 5 we explain how service configuration is preceded by the first sub-process, to complete the *serviguration* process. We

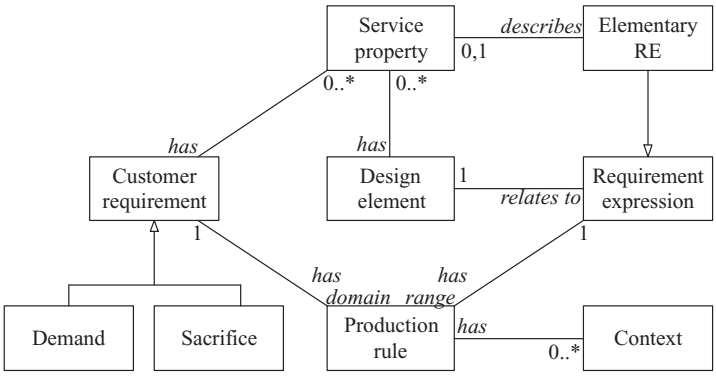

Figure 3.14: Production rules for transforming customer terminology to supplier terminology

include in the service ontology several constructs to support the transformation process between the service value (customer side) perspective and the service offering (supply-side) perspective. These are listed below, and depicted in Figure 3.14.

Production rule. Statements of the form "if a particular situation X is encountered then select solution Y" are referred to as production rules. In its simplest form, in our case a production rule would be "if customers have demand X, offer them resource Y", meaning that resource Y presents a benefit that can satisfy demand X. As we will see in Chapter 5, we use several types of production rules to reason about the suitability of resources for a given demand.

Attributes. A production rule is described by the attribute 'has name'.

Relations. A production rule has two parts: the IF part, to which we refer as 'domain', and the THEN part, to which we refer as 'range'. The domain of a production rule is either a demand (if the requirement relates to a desired outcome) or a sacrifice (if the requirement specifies an acceptable sacrifice). Demands and sacrifices are referred to as *customer requirements*. Conceptually, the range of a production rule is a resource. However, computationally, a production rule is related to a *requirement expression*, which describes a requirement in supplier terms. A requirement expression, in turn, is related to a design element, the supertype of a resource. Therefore we model the two relations "production rule hasDomain... customer requirement (cardinality 1)" and "production rule hasRange... requirement expression (cardinality 1)". As can be seen in Figure 3.14, every customer requirement and every design element can be described by zero or more service properties. Consequently, a production rule may be of the type "if customers have demand X, offer them resource Y" or of the type "if customers have demand X with service properties L, M,... N, offer them resource Y with service properties A, B,... C". A production rule also has the relation "production rule has context", which we explain below.

Visualization. We discuss the visualization of production rules in Chapter 5.

Example. If customers have the demand "be able to pay while I am abroad", and the context is "customer type is business man", offer them resource "credit card" with service property "esteem: high".

Context. The applicability of a production rule may depend on various factors. For example, two different customers who have the same demand for payment facilities may require a different solution to satisfy their demand, because of *their* characteristics or because of characteristics *of the available solutions (resources)*. Take for example a service that is provided only in a limited geographical area. The benefits (resources) of this service should be used as a solution for customer demands only if the context of this service request specifies that the geographical restriction is valid. Another example is a customer who has a demand for medical assistance. Different benefits (resources, outcomes of services) will be offered to this customer during daytime and at night. Hence context knowledge acts as a filter, to decide which knowledge pieces must be taken into account in the search for a solution for a customer demand (Brézillon 1999).

Similarly, also a whole service element may be subject to some context information. For example, certain services are provided only to the elderly, so a minimum age is required for the consumption of such services. This is facilitated by the relation 'service element has context'.

Attributes. A context is described by the attributes 'has name' and 'has expression'.

Example. Context 'customer type' with expression 'business man'; context 'location' with expression 'zip code 7100-7199'; context 'time' with expression '24x7'.

3.6 Summary

In this chapter we presented the concepts and relations that constitute our *servigura-tion* service ontology. We split the ontology into two perspectives, a supplier perspective and a customer perspective, so that on the one hand the actual service offerings of various suppliers can be compared and combined into service bundles, and on the other hand, any service or service bundle presents customer value, and can be offered when specific customer needs occur.

The service offering perspective of the ontology can be used to model real-world services. The concepts that constitute this perspective describe services as economic activities in which customers and suppliers exchange economic values (see Section 3.3.1), and at the same time also as a type of *components* (see Section 3.3.2), so that services can be configured using the same algorithms that are used for component configuration in configuration theory, once constraints (see Section 3.3.3) and

requirements for the configuration process (see Section 3.3.4) are defined . In Chapter 4 we show how we use all these constructs to actually configure service bundles out of more elementary services.

In the description of concepts and relations that constitute the service ontology we did not handle inherent domain-related constraints on how to use these concepts and relations. Yet, these constraints must be formalized for the realization of information systems for service bundling. Therefore we present a list of these constraints, using a formal notation, in Appendix A.

Our ontology distinguishes itself from traditional business research and traditional computer science research. Unlike traditional business research we employ computer-based knowledge representation techniques, such that domain knowledge becomes computational. We distinguish ourselves from research in computer science in several aspects.

First, content wise our knowledge modeling adheres to principles from business research, and our ontology explicitly incorporates business logic into a computational framework. This is opposed to other research efforts that often focus on computational frameworks for *realizing* activities, and do not *reason about the higher-level need* for these activities. Configuration, and in particular also service configuration (in the context of semantic web services) has been performed by others in the computer science research community, but business logic has not been integrated into this research. Our ontology integrates service configuration and business modeling into service modeling.

Second, we do not require users to specify their bundling requirements in supplier terminology. Therefore our ontology includes also a customer perspective next to the traditional supplier perspective.

Third, besides a computational framework for software-aided processes, we represent our models also graphically, to enhance communication with stakeholders.

Fourth, as opposed to traditional research on configuration within computer science, our ontology aims at configuring *intangible* elements, rather than physical goods. We will discuss this issue more elaborately in the next chapter.

Chapter 4

Service Offering Perspective: Service Bundling as a Configuration Task

***Note**: In this chapter we show how service bundling can be represented as a task of configuring intangibles and tangibles, from a supplier's perspective. We use our service ontology and a configuration ontology to show how services can be considered as components, and a service bundle as a configured component. This chapter is based on an article in the IEEE Intelligent Systems magazine (Akkermans, Baida, Gordijn, Peña, Altuna & Laresgoiti 2004) and on a configuration ontology that was developed by Altuna, Cabrerizo, Laresgoiti, Peña & Sastre (2004).*

In this chapter we show how service bundling can be represented as a configuration task. Configuration traditionally deals with tangible artifacts. Services, on the other hand, are of an intangible nature. Nevertheless, we show that a service bundling task can be represented as a configuration task, so that known configuration algorithms can perform the task. We achieve this by carrying out the following steps:

1. We use an ontology of services that we have developed based on business research and practice.
2. We map this service ontology onto an existing ontology of configurations, to represent services as components, and a service bundling task as a configuration task.
3. We employ existing configuration algorithms that are based on the latter ontology to configure service bundles.

4. We use a mapping in the reverse direction than in step 2 – from configurations to services – to interpret and represent configuration solutions as service bundles.

Traditionally, the configuration of physical goods is facilitated by describing physical *interfaces* of components (Borst 1997). These interfaces determine whether one component can be connected to another. For example, an electricity socket has a certain physical interface, to which only electricity plugs that adhere to that interface can be connected. This may sound rather basic, but when dealing with services instead of with electrical appliances, it is often impossible to describe the elements of a configuration (e.g., socket, plug) using physical characteristics, because services do not have physical characteristics as electrical appliances do. Nevertheless, our service ontology, described in the previous chapter, retains the principle of connecting components based on their interfaces. The interfaces of a service describe the exchange of money, capabilities and other objects of economic value between customers and suppliers.

After shortly discussing configuration research in Section 4.1, in Section 4.2 we present a generic configuration ontology. Next, in Section 4.3 we describe the mapping of concepts from the service ontology onto concepts of the configuration ontology, to facilitate service configuration. After shortly discussing service bundles in Section 4.4, in Section 4.5 we describe the algorithm we use for configuring services. We conclude this chapter with a summary in Section 4.6.

4.1 Configuration

4.1.1 Configuration Task Revisited

The definition of a configuration task that we use in this thesis is borrowed from Mittal & Frayman (1989):

> "Given: (A) a fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at that port, and other structural constraints; (B) some description of the desired configuration; and (C) possibly some criteria for making optimal selections.[1]
> Build: One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the

[1] In our work on service configuration we do not consider optimality criteria; instead, we are interested in all possible solutions.

> connections between the components in the set, or detect inconsistencies in the requirements."

Configuration is a type of design (Wielinga & Schreiber 1997). A design process is described by Motta (1999, pg. 82) as "a function which takes needs, desires, constraints and building blocks as input, and produces an artefact as output". Unlike in generic design tasks, at the start of a configuration design task it is assumed that there is complete knowledge about the existing 'building blocks' (Motta 1999) of which solutions may be built, about requirements and about constraints on how these 'building blocks' can interact (Motta 1999, Wielinga & Schreiber 1997, Sabin & Weigel 1998).

The space of solutions of a configuration design task includes all possible assemblies of components ('building blocks'). The subset thereof that satisfies the configuration's constraints is referred to as *valid configurations*. If valid configurations satisfy also the requirements for the desired configurations, they are referred to as *suitable configurations*. And finally, suitable configurations that satisfy also optimality criteria are called *optimal configurations* (Löckenhoff & Messer 1994, ten Teije et al. 1998, Wielinga & Schreiber 1997).

4.1.2 Product Configuration

We use configuration to bundle services; we configure services. Configuration has been used broadly for so-called "product configuration", the routine configuration engineering activity in the sales-order-delivery process (Soininen et al. 1998). Businesses use automated product configurators to create customer-tailored configurations of their products.

An early product configuration system was the R1/XCON (McDermott 1982), a rule-based configurator of VAX-11 computer systems. Other product configurators were MICON (a system for configuring single-board computer systems) (Birmingham et al. 1988), VT (a configurator of elevator systems) (Gruber et al. 1996) and Cossoack (a system for configuring PCs) (Mittal & Frayman 1989). A list of commercial configurators is given in Sabin & Weigel (1998). Product configurators can use rule-based reasoning, model-based reasoning or case-based reasoning (Sabin & Weigel 1998).

Product configuration systems use various approaches to describe a configuration. Soininen et al. (1998) distinguish between a connection-based approach (Mittal & Frayman 1989), a resource-based approach (Heinrich & Jüngst 1991), a structure-based approach (Cunis et al. 1989) and a function-based approach (Najmann & Stein 1992).

As we show in this chapter, research on product configuration can be used for configuring services if a new layer is introduced on top of this research. The need for a new layer stems from the differences between (tangible) goods and (mostly intangible) services. Although the term 'product' encompasses goods as well as services, in fact product configurators are *goods* configurators. R1/XCON, MICON and Cossoack configure computers; VT configures elevators. Due to their tangible nature, these can be described unambiguously by their physical characteristics. This terminology is typically a supplier terminology (a customer who wishes to configure a PC needs to specify his requirements in terms of the available components that shall be part of a (configured) PC). As we argued in Section 2.4.2, services differ from goods in their intangible nature, due to which they cannot be described in unambiguous terms, based on observable physical characteristics. Various persons may experience a service differently, and the supplier perspective on services may differ substantially from the customer perspective. Furthermore, because no two services are the same or are experienced as the same (due to the influence of service personnel, of time and location, of customer expectations and more), and because different service suppliers have different strategies (for bundling, pricing and more), describing rules for connecting services is not as straightforward as it is with physical goods, where every good can be identified by a unique ID (e.g., a model number), even when sold by various suppliers.

As we show in the rest of this chapter, the service ontology presented in Chapter 3 can be used as a layer on top of a configuration ontology that is based on the same configuration theory as used for product (goods) configurators. The service ontology adds the required mechanisms to describe services by means of the value exchange between customers and suppliers. We show that concepts of the service ontology have equivalent concepts in a configuration ontology, so that the latter can be used for service configuration. Developing a configuration ontology itself is beyond the scope of our work. Instead, we use a configuration ontology that our Spanish project partner *Fundación LABEIN* developed, based on established configuration literature. In the next section we present this configuration ontology, developed and published by Altuna et al. (2004).

4.2 Configuration Ontology

In order to come to a generic configuration ontology, it was important to understand the meaning of configuration. The most commonly used definition of the configuration task was given by Mittal & Frayman (1989) (see Section 4.1.1). Based on this definition, we derive that a configuration (the result of a configuration task) requires:

1. A set of components, such that these components can be described by a set of

properties and ports connecting them to other components.

2. Constraints at each port that describe the components that can be connected at that port, as well as other structural constraints.

3. User requirements with the description of the desired configuration, and possibly some criteria for optimizing the solution.

Therefore, a configuration ontology has to address issues related to the following concepts: components, ports, connections, properties or attributes, constraints and user requirements.

Configuration has been recognized as a topic of research for several decennia. Most definitions of a configuration task found in the literature are a slight variant of the first generic definition given by Mittal & Frayman (1989), as quoted above. Further research was performed by Gruber et al. (1996), Wielinga & Schreiber (1997), Soininen et al. (1998), Schwenzfeger et al. (1992), Günter (1992), Heinrich (1991), Heinrich & Jüngst (1991), Kopisch & Günter (1992), Schwanke & Benert (1990), Schweiger (1992), Tank (1992) and more.

The configuration ontology presented here is based on the above literature. It has been divided into three sub-ontologies that represent different aspects of the knowledge that is necessary for the specification of well-formed configuration problems.

- The **Configuration Components Sub-ontology** contains all the static information, and therefore provides the basic taxonomy definitions to define configuration problems, e.g., components, attributes and relations.

- The **Configuration Constraints Sub-ontology** contains information to describe constraints. These constraints can apply to the components or to the desired configuration and are associated with the Components sub-ontology and with the Requirements sub-ontology respectively.

- The **Configuration Requirements Sub-ontology** provides taxonomies for describing the user requirements specification. This is the input/output description of a system that performs a configuration task.

In their configuration ontology that we use here, Altuna et al. (2004) distinguish between high-level configuration problems and detail-level configuration problems. The former concern *which* components are grouped into a complex component. The latter concern *how* components are connected within a complex component, and involve arrangement such that the components are connected through their ports. Both types of configuration problems can be applied to tangible products (e.g., personal computer), called *goods*, or to intangible products (e.g., medical treatment) called *services*.

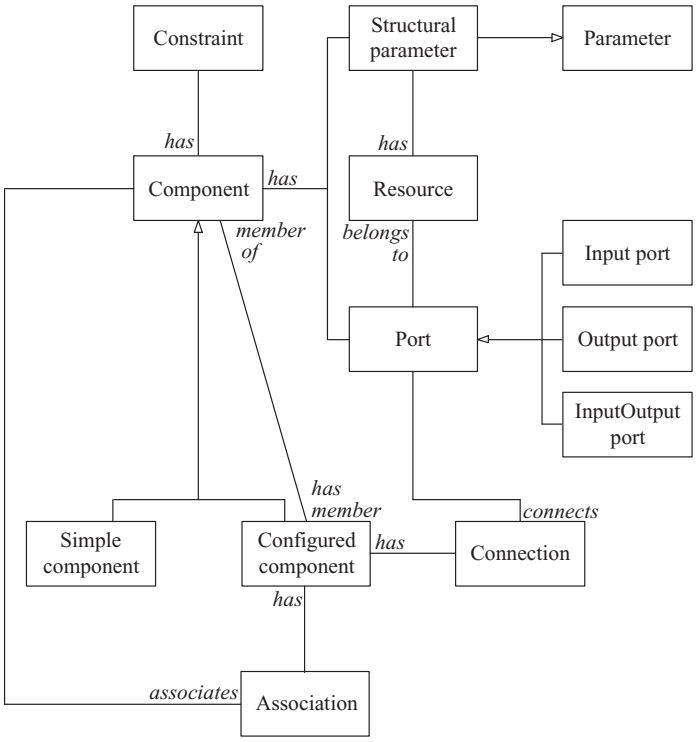

Figure 4.1: Components sub-ontology

The three sub-ontologies are described hereunder. We do not present the whole configuration ontology, but only those parts that we use to configure services. Mainly, as the current version of the service ontology does not include optimality criteria (instead, we are interested in *all* possible solutions), the concept 'optimality criteria' is not being discussed hereunder.

4.2.1 Components Sub-ontology

Figure 4.1 presents a visualization of the components sub-ontology. The various concepts of this sub-ontology are discussed below (the figure uses the same UML class diagram notations as explained in the legend of Figure 3.2).

Component. A Component can be a primitive module (SimpleComponent) or an assembly of primitive modules (ConfiguredComponent) that participate in configuration design. Components can be physical goods or may also represent functional and behavioral abstractions, i.e., services. A component may have constraints, parameters and ports, and it may be a member of a configured component.

Simple Component. A Simple Component is the most basic (atomic) component

that participates in configuration design. As a subtype of Component, it inherits all the properties of Component.

Configured Component. A Configured Component is an assembly of Components (SimpleComponents and/or ConfiguredComponents) that can be linked via Associations (high-level configuration) or via Connections (detail-level configuration). Components that are associated with a configured component are referred to as "members" of a configured component. Therefore a ConfiguredComponent is the solution of a performed configuration problem, but can play a role as a Component to be configured in a new configuration problem.

As described in the configuration literature, components have constraints, parameters and ports (Wielinga & Schreiber 1997). As such, we can identify these concepts in the Components sub-ontology.

Structural Parameter. Structural parameters describe characteristics of components (simple ones or configured ones) or of resources. Like other parameters, they are described by the attributes 'has value', 'has value type' and 'has unit'. Structural parameters play an important role when performing high-level configuration (configuration design involving parametric constraint satisfaction, discussed in Section 4.5).

Port. Ports constitute the interface of components to the external world, i.e., ports specify how components can be connected. They play an important role when performing detail-level configuration (configuration design involving arrangement, see Section 4.5). The ontology defines three sub-concepts of port in order to distinguish ports that receive something (from other ports) from ports that give something (to other ports). The former type is called InputPort, and the latter type is called OutputPort. The term InputOutputPort is reserved for ports that can at the same time give and receive something.

Resource. This concept stores the knowledge about what ports are exchanging. A Resource stores also information about how the port to which it belongs can be interconnected to other ports.

Connection. The Connection concept arises as reification of the connection relationship between two ports. The Connection is related to ports and therefore to the detail-level configuration problem type.

Association. This concept indicates that a Component is member of a Configured Component. The Association concept arises as reification of the connection relationship between two Components. The Association is related to Components and therefore to the high-level configuration problem type.

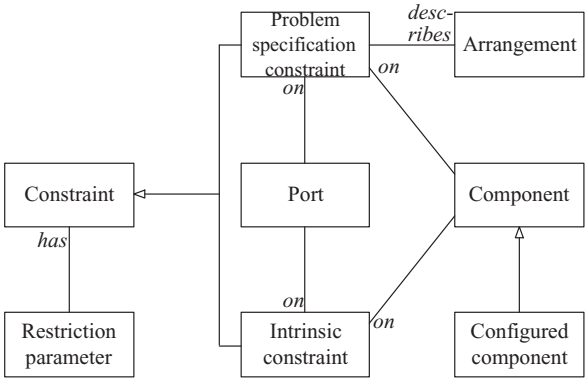

Figure 4.2: Constraints sub-ontology

4.2.2 Constraints Sub-ontology

This sub-ontology (see Figure 4.2) contains information regarding the constraints applicable over the domain. The constraints represent specific domain knowledge, and therefore will be applied over the instances of the specification domain.

Intrinsic Constraints. These constraints are applicable to the Components sub-ontology. Such constraints are inherent to the domain, and will be applied for any user requirements. Once these constraints have been applied, a set of "valid solutions" is returned (Löckenhoff & Messer 1994, ten Teije et al. 1998, Wielinga & Schreiber 1997).

Problem Specification Constraint. This concept indicates the constraints that stem from the Requirements sub-ontology, after user requirements have been transformed into well-specified constraints that describe the kind of solution requested by the user. These constraints restrict the set of valid solutions to the set of all "suitable configurations" (Löckenhoff & Messer 1994, ten Teije et al. 1998, Wielinga & Schreiber 1997).

Restriction Parameter. Restriction parameters parameterize problem specification constraints by specifying parameter values, related to the constraints.

The configuration ontology uses the following typology of constraints:

- *Unary Constraints* limit the possible values of a configured component member.
- *Binary Symmetric Constraints* are applicable to two components in the same way (e.g., A=B).

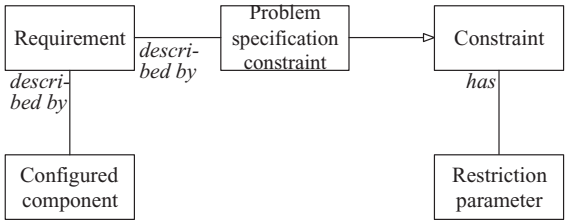

Figure 4.3: Requirements sub-ontology

- *Binary Non-Symmetric Constraints* are applicable to two components having different roles. One component acts as a main actor, and the other as a secondary actor (e.g., *if A then B*).
- *Incremental constraints* are incrementally applicable to a configured component as a whole (rather than to a component within the configured component), when some of its members have been assigned a concrete value.
- *Global Constraints* are applicable to a configured component once the rest of the constraints have been satisfied, and all of its members have been assigned a concrete value.

4.2.3 Requirements Sub-ontology

This sub-ontology (see Figure 4.3) contains the concepts and relations necessary to describe the desired configuration.

Requirement. The customer requirements specify the characteristics of the configured component that the user wishes to obtain. They are described by well-defined Problem Specification Constraints. Customer requirements are in fact a description of a desired configured component, which has to be designed. This is facilitated by the relation describedByComponent.

Restriction Parameter. The objective of this parameter is to parameterize the constraints previously defined in the Constraint sub-ontology by specifying parameter values, related to the constraints.

4.3 A Transformation Between Service and Configuration Ontologies

In Section 3.3 we presented the service offering perspective of our *serviguration* service ontology. Its constructs enable us to model services, business rules for cre-

ating service bundles and bundling requirements in terms of the desired inputs and outcomes of a service bundle. Next, in Section 4.2 we presented a configuration ontology that can be used by configuration algorithms to configure components out of more elementary components. In order to use such algorithms for service bundling, we need to show how the configuration ontology is suitable for describing services. We do that by mapping concepts and relations between the two ontologies. To put it differently, we need to show how knowledge modeled using the service ontology ('service knowledge') can be transformed into knowledge modeled by the configuration ontology ('configuration knowledge'), such that (1) a problem stated in service terminology can be expressed in terms of the configuration terminology, and (2) when this configuration knowledge is used to configure configured service components, the result will be a suitable solution for the problem posed by the service knowledge.

This transformation process is facilitated by a mapping between concepts and relations of the two ontologies. We prefer to refer to it as 'transformation', rather than *ontology mapping*, because as Kalfoglou & Schorlemmer (2003) show, it is "arguably impossible" to provide standardized definitions and scope of the term *ontology mapping*, and providing such a standardized definition is not the aim of this thesis. Our transformation is a mapping between concepts and relations of two ontologies, referred to as "ontology mapping in terms of morphisms of ontological signatures" by Kalfoglou & Schorlemmer (2003).

In the rest of this section we describe how concepts in the service ontology are mapped onto concepts in the configuration ontology. This facilitates the transformation of service knowledge into configuration knowledge. Naturally, a transformation is required (and has been performed) also in the opposite direction (a mapping between two concepts or relations is not necessarily symmetric), so that solutions (configured components) can be represented as service bundles. As the configuration ontology includes three sub-ontologies, we divide our discussion into three parts.

4.3.1 Components Sub-ontology: Services are Components

The main idea behind using a configuration ontology for service configuration is that services can be described in accordance with component definition, and a service bundling task as a configuration task. Accordingly, a service element is a component, an elementary service element is a simple component, and a service bundle is a configured component. Table 4.1 presents an overview of mapping service concepts and relations onto constructs of the components sub-ontology. As can be seen from the table, the concepts *input interface* and *outcome interface* in the service ontology have no explicit equivalent in the configuration ontology. This is resolved on the port level: service ports that belong to an input interface are equivalent to input ports in the configuration ontology, and service ports that belong to an outcome interface are

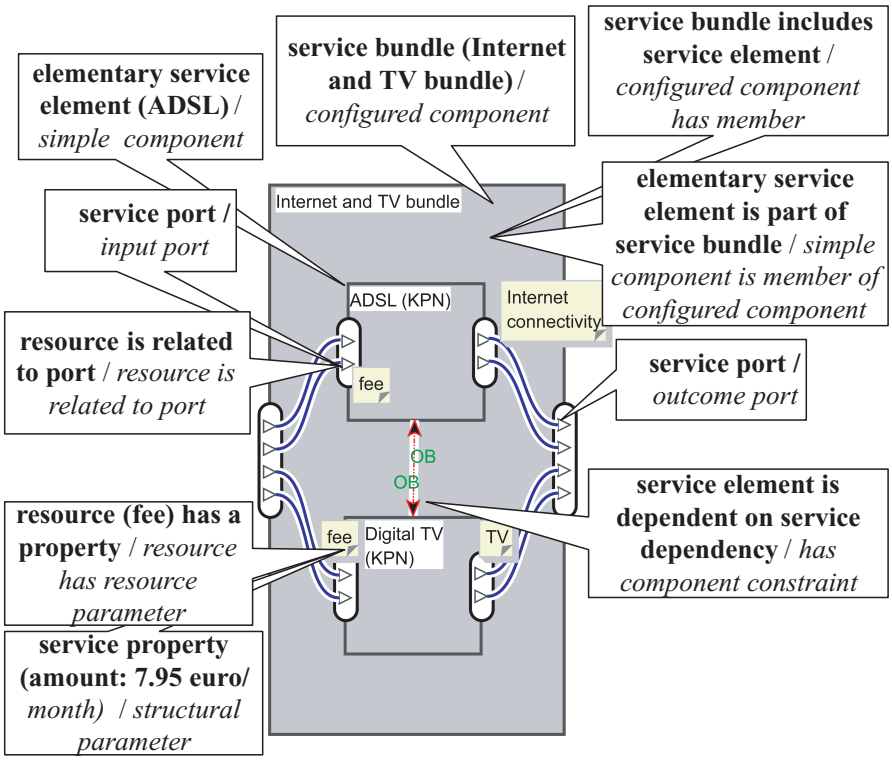

Figure 4.4: Mapping service concepts and relations with concepts and relations of the components sub-ontology. Concepts from the service ontology appear in **bold** text, and concepts from the configuration ontology appear in *italic* text.

equivalent to outcome ports in the configuration ontology. The mapping presented in Table 4.1 is exemplified by a visualization of a service bundle in Figure 4.4.

4.3.2 Constraints Sub-ontology: Intrinsic Constraints

The service ontology includes two explicit types of constraints (service dependencies and conditional outputs), and an implicit type of constraints.

Service dependencies in the service ontology describe constraints for combining service elements into a service bundle. Every one of the six possible service dependencies is expressed as an intrinsic binary constraint in the configuration ontology:

1. Binary constraint CoreEnhancing: Component (service element) B is enhancing another component A (and thus A is a core component of B).

Table 4.1: Mapping service concepts and relations with concepts and relations of the components sub-ontology

Service ontology	*Configuration ontology (components sub-ontology)*
Elementary service element 1. is part of service bundle[*1] [*2] 2. has a property[*1] [*2] 3. is dependent on service dependency[*1] [*2] 4. is dependee of service dependency[*1] [*2] 5. has an input interface[*1] [*2] 6. has an outcome interface[*1] [*2]	Simple component 1. is member of configured component[*1] 2. has component parameter[*1] 3. has component constraint[*1] 4. has component constraint[*1] 5. (no equivalent) 6. (no equivalent)
Service bundle 1. includes service element	Configured component 1. has member
Resource 1. is related to port 2. has a property[*1] 3. is composable 4. is consumable	Resource 1. belongs to port 2. has resource parameter 3. is composable 4. is consumable
Service port (when the range of the relation 'is part of interface' is an instance of *input interface*)	Input port
Service port (when the range of the relation 'is part of interface' is an instance of *outcome interface*)	Output port
Service property	Structural parameter

[*1]Inherited from supertype.

[*2]This relation is valid for all service elements (elementary and bundles).

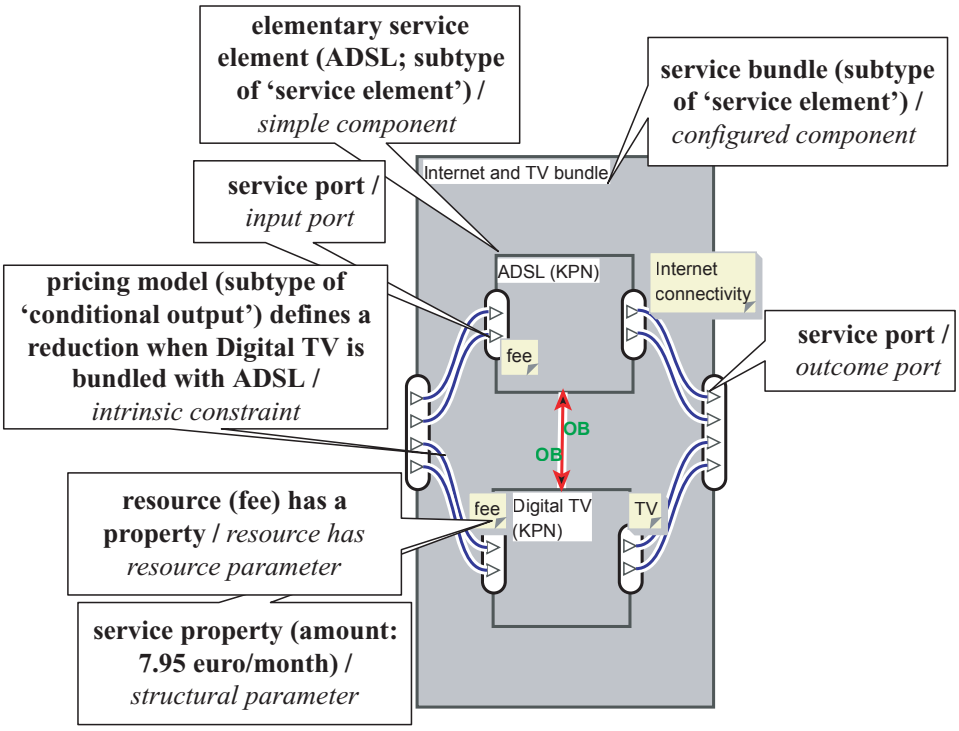

Figure 4.5: Mapping service concepts and relations with concepts and relations of the constraints sub-ontology. Concepts from the service ontology appear in **bold** text, and concepts from the configuration ontology appear in *italic* text.

2. Binary constraint CoreSupporting: Component (service element) B is supporting another component A (and thus A is a core component of B).

3. Binary constraint OptionalBundle: Component A may be bundled with another component B.

4. Binary constraint Bundle: Component A has to be bundled with another component B.

5. Binary constraint Substitute: Component B can replace component A.

6. Binary constraint Excluding: Component B cannot be associated with the same configured component with which also component A is associated.

Conditional outputs describe constraints that determine the value of a property of a resource, possibly based on other properties of other resources, or even of the same resource.

Table 4.2: Mapping service concepts and relations with concepts and relations of the constraints sub-ontology

Service ontology	*Configuration ontology (constraints sub-ontology)*
Service element	Component
Resource 1. has a conditional output[*1]	Resource 1. port has port constraint (in the configuration ontology the constraint is assigned to the port of the relevant resource)
Service property	Structural parameter
Conditional output 1. has a formula 2. has a property domain 3. has a property range 4. assigned to port	Intrinsic constraint 1. has constraint expression 2. constraint on main component 3. constraint on component 4. constraint on main port
Pricing model 1. assigned to port[*1]	Intrinsic constraint 1. constraint on main port
Service port 1. port has a conditional output	Port 1. has port constraint

[*1]Inherited from supertype.

Implicit constraints are inherent to services in general, and are therefore not modeled explicitly in the service ontology. They can be expressed by intrinsic constraints in the configuration ontology. They include the following constraints. A full list is given in Appendix A.

1. A unary constraint to define when two resources are considered equal.
2. A unary constraint to define when one resource is considered bigger than another resource.
3. An incremental constraint that enforces that no cycles occur in a model (services providing resources for themselves). It follows an algorithm based on the transitive property of a connection and the interconnection of the input ports of a service element to the outcome ports of the same service element.
4. A global constraint that removes uninteresting solutions. If an available resource is not consumable, and it is required X times within a bundle, all solutions in which it is being used less than X times in the bundle, are not interesting, because they require extra, unnecessary resources to fill in the gap that the first resource could fill with no extra costs. These solutions should therefore be discarded.

Table 4.2 describes the relevant mappings between the two ontologies. This mapping is exemplified by a visualization of a service bundle in Figure 4.5.

4.3.3 Requirements Sub-ontology: Resources and Service Properties are Restriction Parameters

The requirements sub-ontology describes a desired configuration. When users specify their requirements, in fact they simulate a service bundle. They are interested in a service bundle that requires certain inputs and provides certain outcomes (other inputs and outcomes are possible too, as long as the required ones are available). Service properties describe the values of these inputs and outcomes. Similarly, the requirements sub-ontology describes a simulated configured component, where constraints specify the desired values of parameters. In most of our discussion we mention only requirements that relate to resources. However, the service ontology relates the concept 'requirement expression' to the concept 'design element', which is the supertype of 'resource' and 'service element'. Consequently, it is also possible to set requirements on service elements, rather than on resources. Although we do not use this option currently, it was implemented with future scenarios in mind, when service properties may be added to services, and they may serve as configuration criteria too. The mapping of concepts and relations, dealing with the requirements

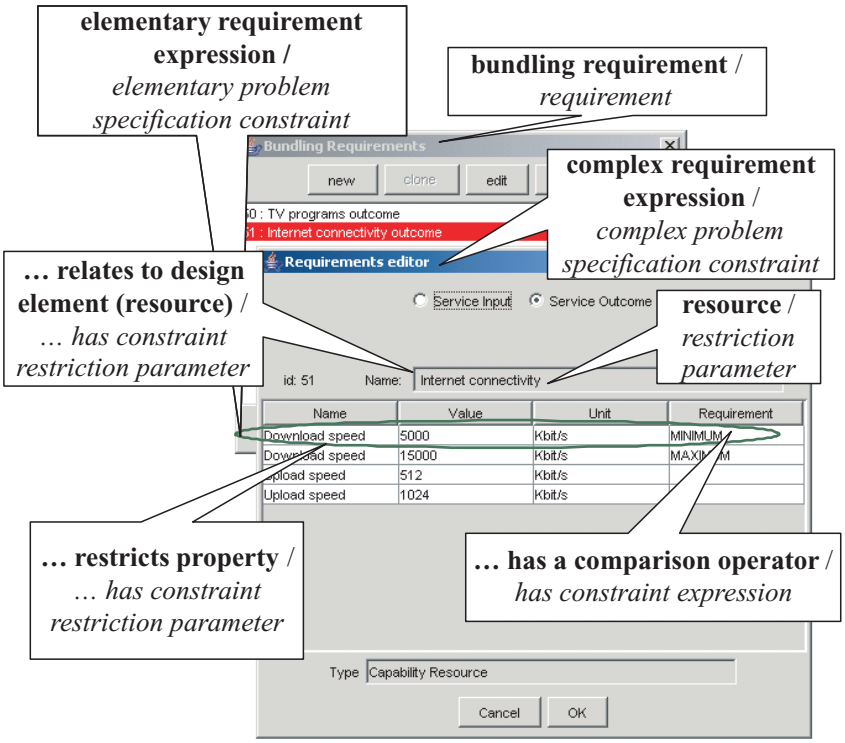

Figure 4.6: Mapping service concepts and relations onto concepts and relations of the requirements sub-ontology. Concepts from the service ontology appear in **bold** text, and concepts from the configuration ontology appear in *italic* text.

sub-ontology, is presented in Table 4.3. The mapping is exemplified by a visualization of a bundling requirement in Figure 4.6, where a customer is interested in a service bundle that provides an Internet connectivity with a download speed between 5000 and 15000 Kbit/s, and a TV subscription.

4.4 Constructing Service Bundles

Figure 4.7 depicts different ways to construct a service bundle from several service elements. Being a composite service element, a service bundle also has an input interface and an outcome interface. These interfaces are identical to the union of all the input and outcome interfaces of the service elements within that bundle. Two exceptions to this general rule exist. First, certain resources can be consumed more than once; that is, they can appear twice or more as inputs of service elements within a service bundle, and yet only once in the service bundle's input interface (particularly, information resources can be used multiple times). Second, when resources have

Table 4.3: Mapping service concepts and relations onto concepts and relations of the requirements sub-ontology

Service ontology	*Configuration ontology (requirements sub-ontology)*
Resource	Restriction parameter
Service property	Restriction parameter
Bundling requirement 1. has requirement expression	Requirement 1. described by (problem specification) constraint
Requirement expression	Problem specification constraint
Elementary requirement expression 1. restricts property 2. has a comparison operator 3. relates to design element (range: resource or service element)	Elementary problem specification constraint 1. has constraint restriction parameter*1 2. has constraint expression*1 3. if the requirement relates to a resource: has constraint restriction parameter*1 to specify a resource's name; and has constraint restriction parameter*1 to specify a resource's type; if the requirement relates to a service element: problem specification constraint on main component*1
Complex requirement expression 1. includes requirement expression	Complex problem specification constraint 1. consists of problem specification constraints

*1 Inherited from supertype.

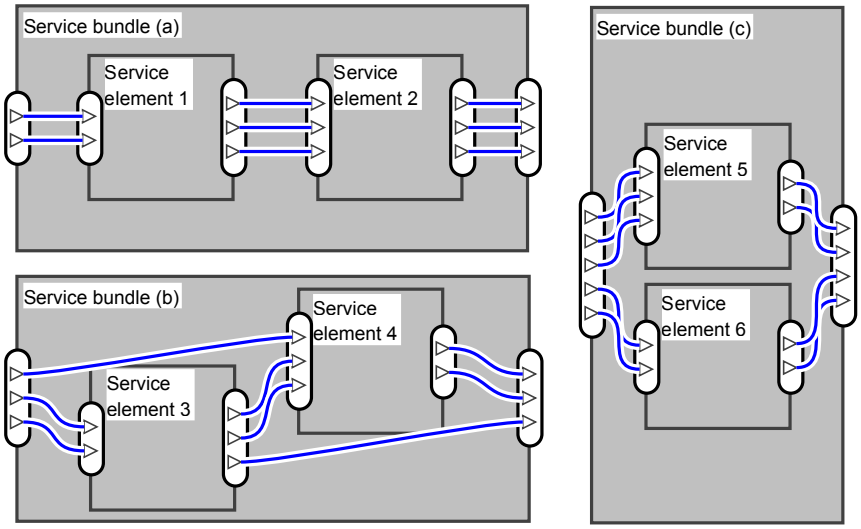

Figure 4.7: Different examples of service bundles: (a) all inputs of service element 2 are provided by service element 1; (b) some inputs of service element 4 are provided by service element 3; (c) no inputs of service element 6 are provided by service element 5.

the compositeness property, we can model multiple resources of the same type as a single resource. For instance, when the input interfaces of two bundled service elements require a fee input, we can compose a single fee resource from these two inputs when designing the bundle's input interface. Often the price for such bundling is lower than the sum of the separate prices.

A service bundle's input interface must provide all the inputs of all that bundle's service elements, unless they are provided internally (one service element might produce an outcome that a different service element consumes as an input). In Figure 4.7, connections (service links) between service ports mean that one service port uses a resource that another service port provides.

Customers consider a service as a bundle of benefits (Kasper et al. 1999). Accordingly, they are typically interested in a black-box view on service bundles, considering the bundle's required inputs and its produced outcomes, and disregarding its internal structure. So, a black-box view is often the most useful view for letting the end customer know whether services satisfy his external requirements. Black-box views do not show the internal arrangement of components, which is the result of the detail-level configuration. Suppliers, on the other hand, are interested in the internal structure of a service bundle: which elementary service elements are included in a bundle, and how they are inter-related. This is shown in a glass-box view, as depicted in Figure 4.7. A glass-box view is obtained after both parts of the configuration algo-

rithm, described in Section 4.5, have been performed (high-level and detail-level).

4.4.1 Service Substitution

Substitution differs from all other service dependencies in the fact that it is computationally redundant, for service configuration. We choose to include it in our ontology nevertheless for two reasons:

1. The ontology serves domain experts for modeling services. Substitution is an important notion for them; it is part of their conceptual model of services. Forcing them not to use it will be unnatural. Conceptually, this notion *belongs* in the service ontology.

2. Incorporating the notion of substitution in the service ontology enables software developers to build a control mechanism to assist domain experts in modeling services (see Appendix A for details).

We will now explain why substitution is computationally redundant for service configuration. In Appendix A we show that substitution has a computational role in building a control mechanism for ontology based software tools.

Two criteria should be taken into consideration in understanding the computational redundancy of the notion substitution for service configuration: what substitution actually means, and why the *serviguration* algorithm adds services to service bundles.

Assume we have two services A and B with a substitution service dependency SU(A, B). Substitution means that (1) the service outcomes of service A are a subset of the service outcomes of service B, or (2) a domain expert thinks that service B can substitute service A (even if the first condition does not hold). Services are added to a service bundle by the *serviguration* algorithm based on two criteria: either (1) because they provide resources that are specified by a bundling requirement, or (2) because of some service dependency. Put together, the two criteria yield the matrix in Figure 4.8.

Case 1: We add service A to an initial service bundle because A provides one or more required resources (say, resource X). Next, service dependency SU(A, B) is triggered, resulting in the creation of a new service bundle, where A is substituted by B. However, since service B provides at least the same resources as service A, the bundle with service B will be generated anyhow, for the same reason that the bundle with A was generated: it provides the required resource X. Therefore modeling substitution between the two services A and B is redundant.

Case 2: We add service A to an initial service bundle because A provides one or more required resources (say, resource X); these are resources which are specified

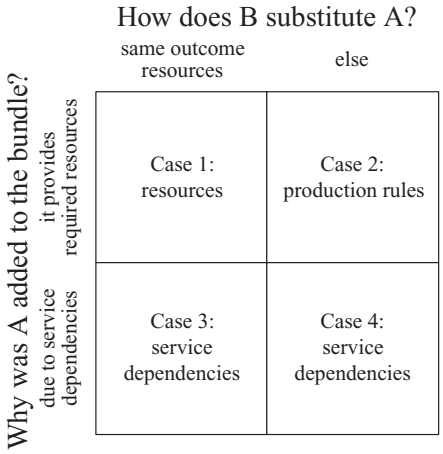

Figure 4.8: Substitution: how is the service dependency SU(A, B) handled?

by *production rules* as good solutions for a given customer demand (this is the first part of the *serviguration* process, explained in Chapter 5). Service B provides other resources (say, resource Y). Yet, domain experts think that B can substitute A. As we describe services by their resources, in this case substitution necessarily means that resources Y can satisfy the same customer demand that is satisfied by resources X. This, in turn, necessarily means that if a production rule exists that triggers the selection of resource X (given a certain customer demand), there must also be a production rule that triggers the selection of resource Y (given the same customer demand). However, if this is the case, service B will be found as a solution also without the substitution service dependency SU(A, B).

Cases 3 and 4: We add service A to an initial bundle because the bundle includes some service C, which has some service dependency that requires (*core/supporting* or *bundled* dependencies) or permits (*core/enhancing* or *optional bundle* dependencies) adding service A to the bundle with service C. In other words, when a bundle includes service C, there is business logic in adding service A to the bundle, together with C. Next, the service dependency SU(A, B) means that there is business logic behind substituting all occurrences of service A with service B. However, this implies that there is business logic behind the combination of services C and B, and therefore a service dependency should be modeled between C and B, that will generate the bundle of C and B. Once again, modeling substitution between the two services A and B is redundant.

The substitution service dependency may therefore be omitted from the configuration algorithm; the same solution bundles will be generated, whether we implement substitution or not. In Appendix A we show that the substitution service dependency has another role, where it is computationally required.

4.5 Configuration Algorithm

Once we know how service knowledge is transformed to configuration knowledge, a next step is to show how a configuration algorithm uses knowledge modeled in terms of the configuration ontology as input to configure configured components. If terminated successfully, a configuration algorithm should provide as output all (zero or more) suitable configured components such that once these are transformed back to service ontology terminology, the computed solutions (service bundles):

- Explicate which service elements are part of the bundle and how they are connected through their service ports.
- Obey all given constraints.
- Satisfy all input customer requirements.

Such a result will prove our claim that services, interpreted as business activities, can be bundled by an automated configuration process in spite of their intangible nature.

To this end, we are interested in a – any – configuration algorithm that meets these requirements. We set no further requirements on the type of algorithm, on its efficiency or on any other quality criteria of configuration algorithms, as this is not required to prove our claim.

Our Spanish project partner *Fundación LABEIN* developed a generic configuration software tool (to be discussed in Chapter 6) based on the configuration ontology presented in this chapter. We successfully used this software tool and its configuration algorithm to configure services, as we shall show in a study in Chapter 7. We present the tool's configuration algorithm below. While we discuss the configuration algorithm in the context of configuring services, the algorithm – as the underlying configuration ontology – is generic, and was used by *Fundación LABEIN* also to configure other components (e.g., physical goods). In fact, the idea to use configuration for service bundling emerged long after *Fundación LABEIN* started to develop a configuration ontology and software tool. As this configuration tool and algorithm were used and were available within the OBELIX project, we used it for our work as well. The configuration algorithm, comprising of a high-level and detail-level configuration, is sound and complete (some solutions are intentionally removed by the algorithm because they are not interesting business-wise, as explained in Section 4.3.2).

The configuration algorithm uses the three sub-ontologies of the configuration ontology as inputs. A components sub-ontology describes components (service elements) and their associated resources (inputs and outcomes). The set of service components to be used by the configuration algorithm is thus known in advance, for two reasons. First, this is required by the configuration definition of Mittal & Frayman (1989) that

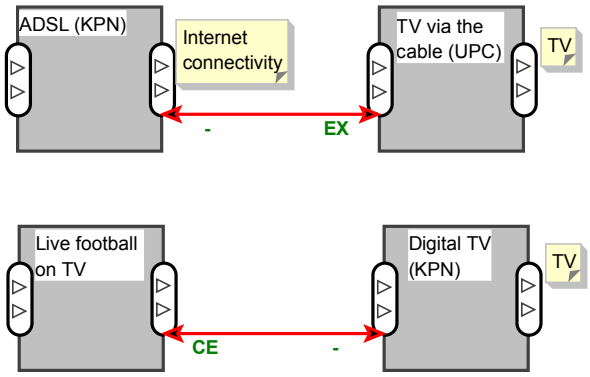

Figure 4.9: High-level configuration algorithm: example service elements. The EX service dependency expresses a business decision of KPN not to market its ADSL service with the TV service of its competitor UPC. The CE service dependency expresses a means of KPN to provide more value to its customers: besides the regular subscription for TV (providing a standard package of channels), football fans can buy a subscription for live football broadcasting.

we use in this thesis. Second, knowledge of all service instances is required in order to define service dependencies (configuration constraints) between components, because service dependencies reflect business logic, which changes per supplier, and sometimes even per service of a single supplier, such that all service components need to be known in advance in order to define constraints. A constraints sub-ontology defines various types of constraints on the configuration process, such as dependencies between service elements, and inherent constraints of the service ontology (e.g., no cycles are permitted). Finally, a requirements sub-ontology describes restrictions on the desired inputs and outcomes to guide configuration. These requirements express the end customer demands by defining the type of resources required and the constraints on their property values.

As explained earlier in this chapter, we distinguish between high-level configuration problems and detail-level configuration problems. The former concerns *which* components are grouped into a configured component. The latter concerns *how* components are connected in a configured component. In the following two sub-sections we present two configuration algorithms that correspond with the two configuration problems. They are executed sequentially. We start with a natural language description of the algorithm, followed by a procedural notation. We use Java pseudocode for the procedural notation, and further explain the algorithms using comments in the pseudocode.

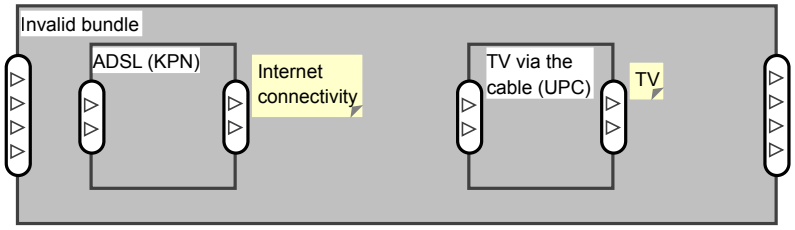

Figure 4.10: High-level configuration algorithm: example of an invalid service bundle (it violates the excluding service dependency).

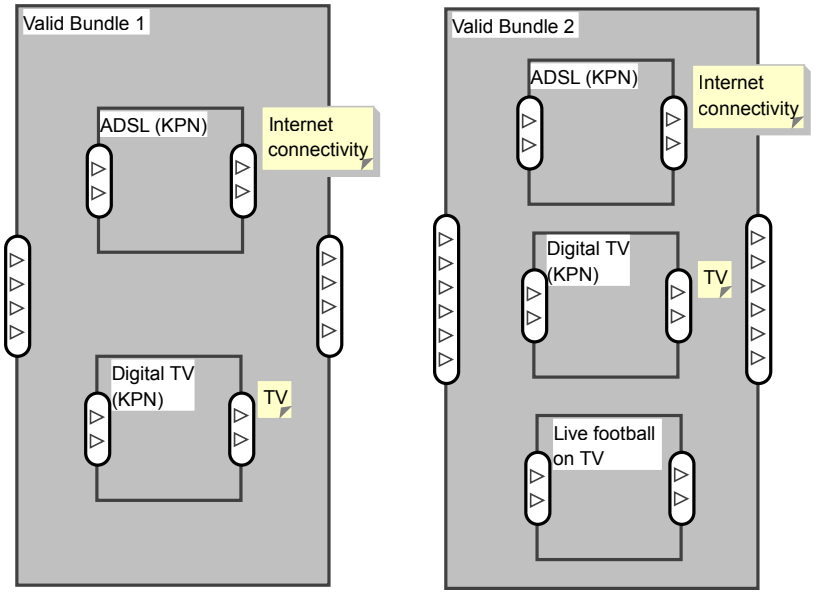

Figure 4.11: High-level configuration algorithm: valid service bundles

4.5.1 High-Level Configuration Algorithm

We explain the algorithm by means of an example. Suppose a user's *bundling requirement* is to obtain two resources: 'Internet connectivity' and 'TV programs'. Let us assume that our library of available services includes the service element 'ADSL' (supplied by KPN) that provides 'Internet connectivity', the service elements 'TV via the cable' (supplied by UPC) and 'Digital TV' (supplied by KPN) that provide 'TV programs', and a fourth service element, 'Live football on TV' (the example does not require knowledge on who supplies this service). All these service elements are visualized in Figure 4.9.

High-level configuration has two steps. The first step consists of identifying initial

elements for the service bundle. Based on the given requirement, the configuration algorithm knows which resources are required ('Internet connectivity' and 'TV programs'). It searches for all service elements that provide the requested resource types, and finds that service 'ADSL' provides 'Internet connectivity' and that services 'TV via the cable' and 'Digital TV' provide 'TV programs' (see Figure 4.9). The second step involves applying service dependencies. Some service elements may require or exclude others. Suppose we have the two service dependencies *excluding(ADSL, TV via the cable)* and *core/enhancing(Digital TV, Live football on TV)* (see Figure 4.9). The first service dependency expresses a business decision of KPN not to market its ADSL service with the TV service of its competitor UPC. The latter is a means of KPN to provide more value to its customers: besides the regular subscription for TV (providing a standard package of channels), football fans can buy a subscription for live football broadcasting. The semantics of the *core/enhancing* service dependency reads: the service 'Live football on TV' may be sold with the service 'Digital TV' to enhance the value of 'Digital TV'; it may not be sold independently of 'Digital TV'. For simplicity, we assume no other service dependencies exist between the four involved services. The first service dependency means that the services 'ADSL' and 'TV via the cable' cannot co-exist in the same service bundle (therefore the service bundle in Figure 4.10 is invalid). So, we derive two possible service bundles: (1) 'ADSL' and 'Digital TV' and (2) 'ADSL', 'Digital TV' and 'Live football on TV' (see Figure 4.11). Each is a good solution. Although the bundle *'ADSL' and 'TV via cable'* provides the desired resources, it is disqualified due to the service dependency *excluding(ADSL, TV via cable)*. The algorithm applies service dependencies to all the considered service elements in a bundle. Whenever the configuration algorithm adds a service element to a bundle, the algorithm applies available service dependencies information again to check the bundle's consistency, resulting in a set of service bundles that are based on business rules and provide the required resource types but do not yet guarantee that all values of resource properties that the customer specified are satisfied. The detail-level configuration algorithm performs this latter task. Hereunder follows a procedural notation of the high-level algorithm:

```
1  Collection configureHighLevel (bundlingRequirements br, serviceLibrary lib) {
2      // This algorithm will generate service bundles. Until the detail level configuration
3      // is done, we cannot be sure that these bundles are indeed good solutions.
4
5      // STEP ONE: Create initial service bundles
6      Collection solutions = initialize an empty Collection of service bundles solutions;
7      // The requirements for the service bundles are a set of resources, possibly with
8      // restrictions on their properties. The restrictions are handled in the detail-level
9      // configuration. Solution service bundles must provide all the specified
10     // resources. If every resource is provided by a different service, the service bundle
11     // will have to include all these services.
```

```
12      Collection solutionCandidates; // Services that are candidates to be included in bundles
13      for (every resource res required by the bundling requirements br) {
14          // Per required resource, we search for services that include that resource.
15          // We do not verify that the found resource has the same properties as specified by
16          // the bundling requirements; this will take place in the detail-level configuration.
17          solutionCandidates = find all services ser in the service library lib, that provide res;
18          // If we found no such services, there is no solution.
19          if (solutionCandidates is empty) {
20              // There is no solution for this configuration problem; solutions is empty.
21              return solutions;
22          }
23          // In case this is the first iteration of a resource res, solutions is still empty,
24          // so we cannot add ser to any service bundle; instead, we first create
25          // empty service bundles.
26          if (solutions is empty) {
27              for (every service ser in solutionCandidates) {
28                  create a new (empty) service bundle sb;
29                  sb.add(ser);
30                  solutions.add(sb);
31              }
32          } else {
33          // In case this is not the first iteration of a resource res:
34          // Any service in solutionCandidates can be added to every service bundle sb in solutions.
35          // Therefore we create solutionCandidates.size() copies of every sb, and add one service from
36          // solutionCandidates to any of these copies. The original can be removed, as it does not
37          // include the desired resource res that all members of solutionCandidates include
38          // (and hence it is not a good solution without the new addition).
39              for (every service bundle sb in solutions) {
40                  create as many copies of sb as solutionCandidates.size();
41                  add these copies to solutions;
42                  add a different service ser from solutionCandidates to every such copy;
43                  remove the original sb from solutions;
44              }
45          }
46      }
47      // STEP TWO: Apply service dependencies within these service bundles
48      for (every service bundle sb in solutions) {
49          solutions = applyServiceDependencies (sb, solutions);
50      }
51      removeDoubles(solutions); // checks for identical solutions
52      return solutions;
53  }
```

```
54 Collection applyServiceDependencies (serviceBundle sb, Collection solutions) {
55     // Apply service dependencies to all members of a service bundle.
56     // Remark 1: This algorithm shows how to handle dependencies with a 1-1 cardinality,
57     // starting at one service and ending at one service; it can be generalized to
58     // service dependencies with cardinality n-n, such that the service dependency has
59     // multiple services as its first argument and multiple services as its second argument.
60     // Remark 2: All service dependencies can be applied consecutively, except for the
61     // Excluding service dependency which also needs to be applied at the end.
62
63     Collection ex =
64             initialize an empty Collection to store all the Excluding service dependencies;
65     // Handle service dependencies per service within the given bundle.
66     for (every service ser in sb) {
67         // Handle all service dependencies that start at this service. Note: a service
68         // dependency can be represented as a function with two arguments (services).
69         for (every service dependency dep starting at ser) {
70             // Identify where the dependency ends.
71             service ser2 = dep.endsAtService();
72             // Now we handle the service dependency based on its type. Six types exist:
73             // Core/Enhancing, OptionalBundle, Core/Supporting, Bundled, Excluding and
74             // Substitute. The latter can be omitted from the algorithm (see Section 4.4.1).
75
76             // Core/Enhancing and OptionalBundle mean that ser2 may be added to the bundle,
77             // but the bundle is a good solution also without the ser2.
78             // Therefore we leave sb as it is, but add a new solution: a copy of sb,
79             // to which we add service ser2.
80             if ((dep is of type Core/Enhancing || dep is of type OptionalBundle) &&
81                     sb does not include ser2) {
82                 create sb1, a copy of sb;
83                 sb1.add(ser2);
84                 solutions.add(sb1);
85                 applyServiceDependencies (sb1, solutions);
86             } else
87             // Core/Supporting and Bundled mean that ser2 must be added to the bundle.
88             // Therefore we add service ser2 to sb.
89             if ((dep is of type Core/Supporting || dep is of type Bundled) &&
90                     sb does not include ser2) {
91                 sb.add(ser2);
92                 applyServiceDependencies (sb, solutions);
93             } else
94             // Excluding means that ser2 may not co-exist in the same bundle with ser.
95             // Therefore if sb includes both ser and ser2, it is not a good solution.
96             if (dep is of type Excluding) {
97                 if (sb.includes(ser2)) { // then sb is an invalid solution
```

```
98                      solutions.remove(sb);
99                      return solutions;
100                 } else {
101                     // Even if sb does not include ser2 yet, ser2 may be added to sb in one
102                     // of the iterations. Therefore we keep a reference to the Excluding
103                     // dependency, and once all iterations have been executed, we re-examine
104                     // all these Excluding dependencies.
105                     ex.add(dep);
106                 }
107             }
108         }
109         // Now verify that there are no two services with the Excluding dependency in sb
110         for (every service dependency dep in ex) {
111             if (sb includes dep.startsAtService() && sb includes dep.endsAtService()) {
112                 // then sb is an invalid solution
113                 solutions.remove(sb);
114             }
115         }
116     }
117     removeDoubles(solutions); // checks for identical solutions
118     return solutions;
119}
```

4.5.2 Detail-Level Configuration Algorithm

In the detail-level configuration we define connections between service ports of the service elements in a service bundle, for every service bundle that the high-level configuration generated. This algorithm connects, as a rule of thumb, as many service ports as possible within a service bundle. The reason to do so is that all inputs that are not provided internally within the bundle must be provided by the customer. If an input can be provided internally, it is unnecessary and disadvantageous to require it from customers. Two service ports of services within a service bundle may be connected if they meet all these requirements:

1. They belong to different service elements within a service bundle.
2. They have different types (if two service elements within a service bundle are connected, the connection must be between an input port and an outcome port).
3. The same resource is assigned to both of them (because a service link between two service ports means that one service port provides a resource for the other service port).

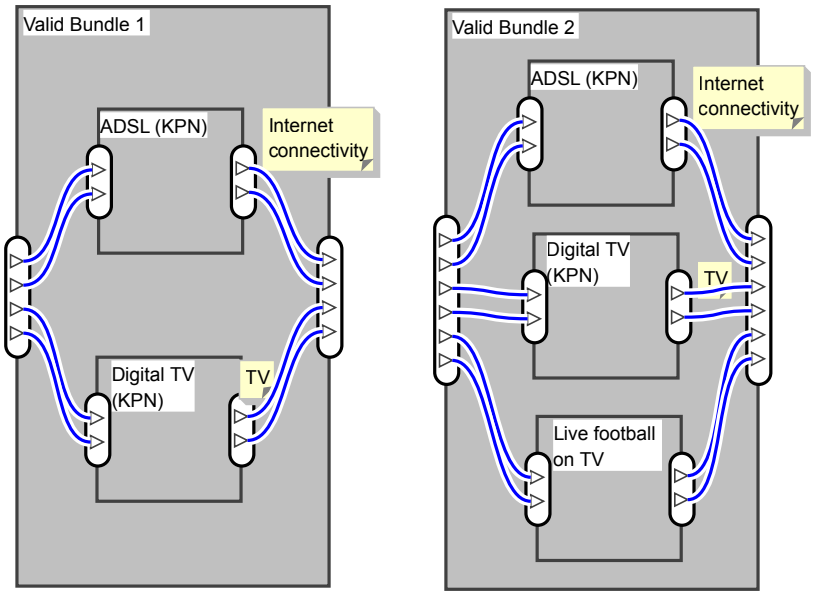

Figure 4.12: Detail-level configuration algorithm: valid service bundles

The configuration engine then checks all restrictions regarding the range of service property values. For example, if the bundling requirement specifies that Internet connectivity is required with a certain, minimum or maximum download speed and/or upload speed, all service bundles that were created during the high-level configuration are tested for compliance with this constraint.

Any input or outcome port that is not connected to other service ports in the same service bundle will appear in the service bundle's input or outcome interface. Any high-level bundle may have multiple detail-level solutions. Figure 4.12 shows examples of the same service bundles as in Figure 4.11, after the detail-level configuration has been performed. Note how service ports are now connected by service links (in this example no service links exist between service elements within a bundle). The input interface and outcome interface of a service bundle provide a black-box view on the service bundle, abstracting away from its internal structure. Hereunder follows the detail-level configuration algorithm in a procedural notation. Throughout the algorithm, whenever service links are added, their validity is checked using constraints that we list in Appendix A.

```
1  Collection configureDetailLevel (Collection bundles, bundlingRequirements br) {
2      // In this algorithm we define connections between ports of the service elements in
3      // a bundle, for every service bundle that the high-level configuration generated.
4      // For brevity, we do not show here how we handle the composability and consumability
5      // properties of service ports and of resources. Instead, here we assume that all
```

```
6   // service ports and all resources are (1) consumable and (2) not composable.
7   // Also, for brevity we do not show here how we handle the case that a high-level
8   // solution may have multiple detail-level solutions. We assume here that every high-level
9   // solution has zero or one detail-level solutions.
10
11  Collection solutions = bundles;
12  // Handle every service bundle separately.
13  for (every service bundle sb in solutions) {
14      // We connect, as a rule of thumb, as many service ports as possible within a bundle.
15      // In other words, if one service in the bundle provides an outcome resource, and
16      // another service in this bundle requires the same resource as an input, and
17      // the service ports of these resources are not yet connected to other service ports,
18      // then we connect (the service ports of) these two resources.
19      // Remark: the following for loop may yield different results when it loops through the
20      // services in sb in a different order. Therefore multiple detail-level solutions are possible.
21      // As mentioned before, here we consider maximum one detail-level solution, so we
22      // do not take the order of looping through sb into consideration.
23      for (every service ser1 within sb) {
24          for (every input resource res1 in ser1) {
25              if (sb includes a service ser2 that provides a resource res2 as an outcome &&
26                  ser1 is not ser2 && res2 ≥ res1 && not res1.getPort().isConnected() &&
27                  not res2.getPort().isConnected()) {
28                  // We can add another control to this if statement, yet it is unlikely to occur.
29                  // In case the outcome resource res2 is required by the bundling requirement,
30                  // and it is consumable, we do not want to consume this resource internally,
31                  // because then it will no longer be available in the outcome interface of the
32                  // bundle, and hence the bundle will not meet the bundling requirements.
33                  // Yet this is not likely to occur, because service dependencies, reflecting
34                  // business rules, are not supposed to create such an inconsistency.
35                  connect the service ports of res1 and res2 with a serviceLink sl;
36              }
37          }
38      }
39      // Now that we have connected as many ports as possible internally, we create
40      // the interfaces of the bundle. All the input ports, respectively outcome ports
41      // (of services within the bundle) that are not connected internally, must appear in
42      // the input interface, respectively outcome interface of the service bundle.
43      serviceInterface inputInterface = sb.getInputInterface().init();
44      serviceInterface outcomeInterface = sb.getOutcomeInterface().init();
45      for (every service ser in sb) {
46          for (every input port p1 in ser) {
47              if (not p1.isConnected()) {
48                  create a new service port p;
49                  assign to p the same resource as assigned to p1;
```

```
50                      inputInterface.add(p);
51                      connect p and p1 with a serviceLink sl;
52                  }
53              }
54              for (every outcome port p2 in ser) {
55                  if (not p2.isConnected()) {
56                      create a new service port p;
57                      assign to p the same resource as assigned to p2;
58                      outcomeInterface.add(p);
59                      connect p and p2 with a serviceLink sl;
60                  }
61              }
62          }
63          // In the high-level configuration we selected services based on the resources that they
64          // provide, but we did not verify that the service properties of the resources meet the
65          // criteria described by the bundling requirements. We do this now.
66          for (every resource res1 required by the bundling requirements br) {
67              if (not (sb includes a resource res2 && res2.getType().equals(res1.getType()) &&
68                      res2.getProperties() satisfy res1.getConstraints(br, res1))) {
69                  // then the service bundle is not a good solution, and it has to be removed from
70                  // the set of solutions.
71                  solutions.remove(sb);
72                  break; // It's a bad bundle.
73              }
74          }
75      }
76      removeDoubles(solutions); // checks for identical solutions
77      return solutions;
78  }
```

Figure 4.13 shows two detail-level solutions for the same high-level solution presented on the left hand side of Figures 4.11 and 4.12. Both services 'ADSL' and 'Digital TV' require a 'fee' input through an input port. In solution 1a, these ports have the attribute 'is not composable', and therefore the two fee inputs cannot be composed into one input fee on the service bundle's interface (instead, the bundle's input interface includes two fee inputs). In solution 1b, on the other hand, the service ports that require fees have the attribute 'is composable', and therefore the two fee inputs can be composed into one fee input on the service bundle's input interface. Naturally, both solutions cannot co-exist for the same configuration problem, because a service port is either composable or not. The decision whether service ports are modeled as composable or not is an important business decision. If they are composable (as in solution 1b), only one bill will be sent to customers. If they are not

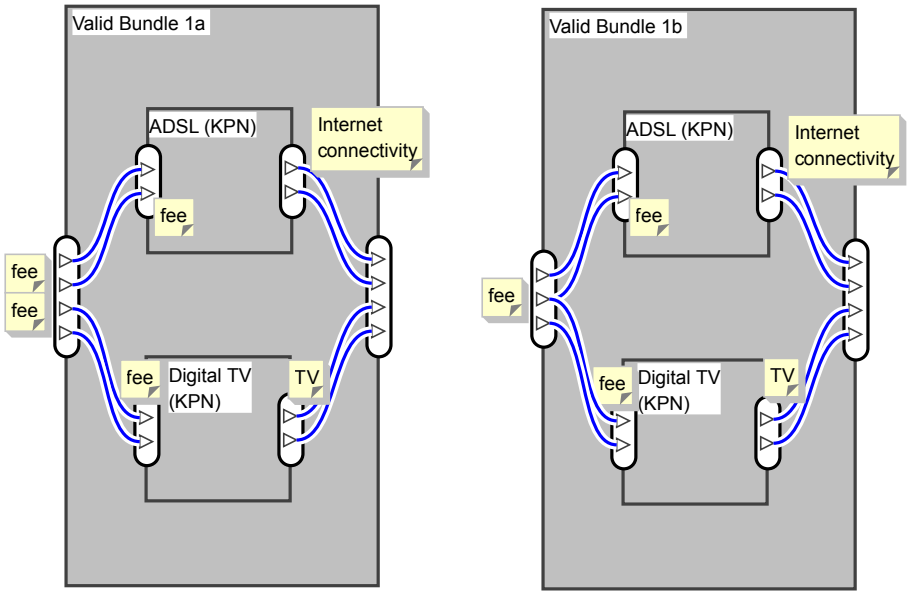

Figure 4.13: Detail-level configuration algorithm: how the composability of service ports influences the construction of service interfaces

composable, customers will receive a separate bill for every service (in reality, KPN issues two separate bills for the two services, as modeled in solution 1a).

4.6 Summary: Configuration Can Handle Intangibles Too

The main claim in this thesis is that services, in spite of their intangible nature, can be configured similarly to (tangible) goods. In this chapter we demonstrated how we achieve this; how we represent the service bundling problem as a component configuration task, which traditionally deals with the configuration of tangibles.

First, we model services using an ontology where services are described as activities in which customers and suppliers exchange benefits, objects of economic value. We also model business rules that tell us how these services can be combined into a service bundle. A desired service bundle is than an assembly of services, such that (1) this assembly requires and/or provides the resources that a user specified; and (2) it has been assembled based on and in accordance with the set of business rules that we had modeled.

Next, to achieve configurability of services, we map concepts and relations of our service ontology onto concepts and relations of a configuration ontology, which is based on traditional configuration research. Services are seen as *components* in this

mapping, and the earlier mentioned business rules are seen as *constraints* on the configuration of components (services).

Finally, once the mapping has been performed, a configuration algorithm can be used to configure components, which are in fact services. We presented a configuration algorithm with which we performed this configuration for real-world, commercial services. This algorithm performs service configuration in the same way as one would configure PCs, elevators, an organization of tables in a room, LEGO blocks or other tangible goods.

In Chapter 6 we discuss the successful implementation of this process in software tools. Chapters 7 and Chapters 8 provide results from two large scale studies in which we performed service configuration successfully to solve business problems.

Chapter 5

Service Value Perspective: Needs Driven Service Offerings

***Note**: This chapter outlines how we add a customer perspective to the service bundling task, so that service bundles satisfy customer demands. It is based on publications in the proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005) (Baida, Gordijn, Sæle, Akkermans & Morch 2005) and in the International Journal of E-Business Research (Baida, Gordijn, Akkermans, Sæle & Morch 2005), and uses our research results from Baida, de Bruin & Gordijn (2003), an article in the International Journal of Web Engineering and Technology.*

In Chapter 3 we introduced the term *serviguration,* encompassing the process of designing customer need driven service bundles. The basic idea behind *serviguration* is visualized in Figure 5.1. The actual service configuration, as presented in Chapter 4, assumes that users can express customer needs in supplier terminology, using concepts from the service offering perspective. This is however typically not the case, as customers and suppliers use different terminologies and have a different view on what customers need (Vasarhelyi & Greenstein 2003).

Consequently, the service value (customer) perspective of our service ontology adds a customers' description of their needs and acceptable sacrifice, which can be satisfied by outcomes of services and inputs of services respectively. This is depicted on the left side of Figure 5.1, where service outcomes and inputs are referred to as resources. In this chapter we discuss how we reason with knowledge modeled by the service value perspective, and how to use that knowledge to derive a description of desired service bundles, using concepts of the service offering perspective. The latter is used as input for the actual service configuration process, described in Chapter 4. Together, Chapters 4 and 5 present the reasoning process that we termed *Serviguration.* The use

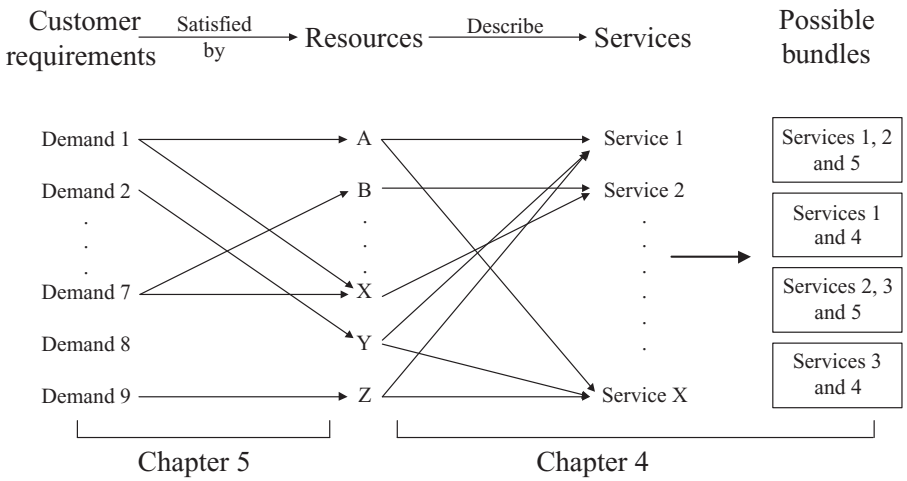

Figure 5.1: Serviguration: configuring service bundles based on customer demands

of the theoretical framework presented in this chapter will be exemplified by a large scale study in Chapter 8.

Two important remarks have to be made:

- While our discussion in this chapter concentrates on deriving a set of desired service outcomes based on customer demands, we use the same mechanisms also to transform the customers' acceptable sacrifice (in customer terminology) to a set of service inputs (in supplier terminology).
- Conceptually, resources provide solutions for demands. Hence we discuss relations between demands and resources. However, due to computational considerations the service ontology relates the concept 'demand', through its supertype 'customer requirement', to the concept 'requirement expression'. The latter is related to a 'design element', the supertype of 'resource'. We explained this is Section 3.5.

The rest of this chapter is organized as follows. We start by discussing in Section 5.1 how to derive a set of concrete customer demands from more abstract wants and needs. Next, in Sections 5.2 and 5.3 we explain how production rules transform customer terminology (demands, sacrifices) to supplier terminology (resources, service outcomes and service inputs). The latter is required as input for the actual service configuration process. Section 5.4 shows how context information, which often is customer-specific, is being taken into account in the *serviguration* process. Finally, Section 5.5 is a summary of this chapter.

5.1 Reasoning about Customer Needs

Services and goods are marketed to satisfy the *needs* of their target groups (Kotler 1988). To put it differently, customers have needs, and these are satisfied by services and goods. Similarly, *serviguration* is triggered by a set of customer needs (modeled in the service value perspective of the ontology), and ends with service bundles (service offering perspective) as solutions for these needs. Needs, wants and demands are the main concepts of the service value perspective, presented in Section 3.4. We repeat their definitions here, as given by Kotler (1988):

- A human *need* is "a state of felt deprivation of some basic satisfaction". As needs are often vague, or abstract, we concretize them by transforming them into more concrete wants.
- *Wants* are "desires for specific satisfiers of deeper needs". Also wants are often not specified in enough detail to find a suitable satisfier. We therefore define demands that describe wants in more concrete terms.
- *Demands* are "wants for specific products that are backed up by an ability and willingness to buy them".

Table 5.1 shows examples of needs, wants and demands, as we modeled in the energy study that we present in Chapter 7. As can be seen from the table, customers specify demands in their own terminology (e.g., 'room heating') or in supplier terminology (e.g., 'telephone line'). The latter happens when customers are already familiar with available services that can satisfy their needs. In our study, the energy utility TrønderEnergi (our partner in the study) wanted to explore possible ways to bundle electricity supply with other (not energy related) services, such that the bundles provide a good solution for customer needs. Therefore, the list of needs, wants and demands presented in Table 5.1 is not complete; it includes only those needs, wants and demands that TrønderEnergi considered to satisfy through existing or new service offerings.

Matching customer needs with available services requires two phases of reasoning:

- If no knowledge exists about the concrete demands of customers, we need to reason about relations between needs, wants and demands, and derive concrete demands based on more abstract needs.
- Once a set of customer demands has been derived, a matchmaking is required between these demands and available service offerings of service suppliers.

In the rest of this section we show how we perform the first of these reasoning processes, and in Sections 5.2 and 5.3 we describe how we perform the second reasoning process.

Table 5.1: Customer needs, wants and demands for the energy utility TrønderEnergi. The notations H/I refer to the customer type: Household or Industrial. Also non energy related needs, wants and demands are included because the study at hand considered bundling energy services with other services.

Customer Needs	*Customer Wants*	*Customer Demands*
Indoor comfort (H,I)	Lighting (H,I); Home services (cooking, washing) (H); Comfort temperature (H,I)	Energy supply (H,I); Hot tap water (H,I); Room heating (H,I); Air conditioning (H,I)
	Energy regulation for budget control (H,I)	Energy regulation for budget control (H,I), with different characteristics (manual / automated; on-site regulation / location-independent
	Temperature regulation for increased comfort (H,I)	Temperature regulation (H,I) with different characteristics (manual / automated, on-site regulation / location-independent)
Social contacts and Recreation (H); Business contacts (I)	Communication (H,I)	Telephone line (H,I); Mobile phone line (H,I); Internet (broadband) (H,I)
Safety (H,I)	Increased security (H,I); Reduced insurance premium (H)	Safety check of electrical installation (H); Internal control of electrical installation (I)
IT support for business (I)	IT-services (I)	ASP-services (I); Hardware (I); Software (I)

5.1.1 Need Hierarchies Comprise of Needs, Wants and Demands

The relation between needs, wants and demands can be described by a hierarchy, "a structure by which classes of objects are ranked according to some subordinating principle" (Stephens & Tripp 1978). Need hierarchies comprise of three levels of aggregation, using the above definitions of needs, wants and demands as a subordinating principle.

Similar hierarchies have been used in the field of Goal Oriented Requirements Engineering (GORE) to transform high-level organizational needs to concrete system requirements (Donzelli 2004). *Needs* capture the answer for the question why a service (either an elementary one or a service bundle) is offered. Similarly, in system/software design *goals* represent why a system/software is needed.

Similar to customer needs, also goals are defined at different levels of abstraction. They capture the various objectives that the system under consideration should achieve (van Lamsweerde 2000, van Lamsweerde 2001). Unlike GORE literature on goal hierarchies (Fuxman et al. 2003), the marketing literature discusses hierarchies (of needs) (Kotler 1988) without providing well-defined relations between elements in the need hierarchies, required for software to reason about needs. We employ GORE techniques to do so.

5.1.2 A Need Graph

The marketing literature only specifies that demands are more concrete wants, and that wants satisfy needs, but it does not describe the logical structure of needs decomposition into wants and of wants decomposition into demands. We fill this gap by introducing AND/(EX)OR refinements. An AND decomposition means that all siblings of a higher-level object (need, want) must be satisfied to satisfy the higher-level object. An OR decomposition means that a higher-level object can be satisfied by satisfying an arbitrary number of its siblings. An EXOR decomposition means that exactly one of the siblings of a higher-level object must be satisfied to satisfy the higher-level object. These constructs can be combined, for example need N1 may be decomposed into wants W1, W2, W3 and W4 as follows: (W1 AND W2) EXOR (W3 AND W4).

We model need hierarchies similar to goal trees. In our case hierarchies are directed graphs, rather than trees, because a demand or want may be related to more than one want or need respectively, so multiple paths may exist between two nodes, which is not allowed in trees. Needs are the top level nodes of the graph; then come wants; and finally demands are leafs. AND/(EX)OR refinements describe the relations between a node in the hierarchy (graph) with related nodes in an adjacent level of the hierarchy. Edges that connect nodes have the semantics "concretized by". This relation does

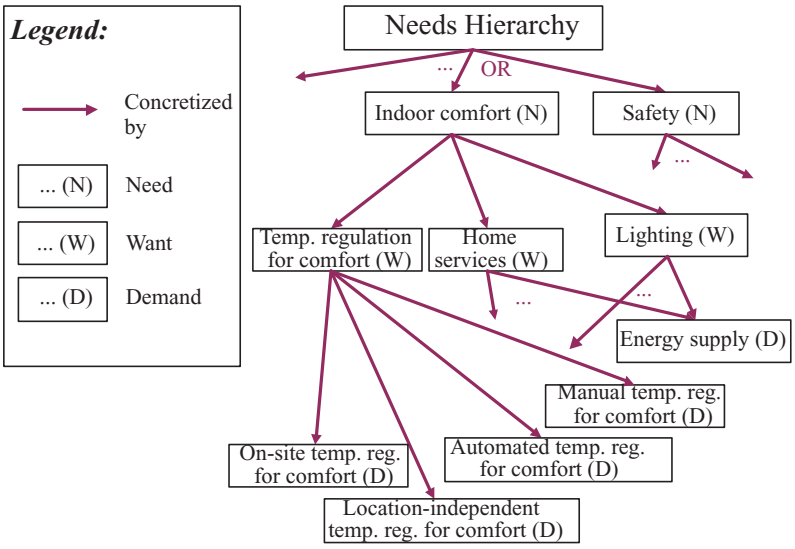

Figure 5.2: Partial need hierarchy from the energy study (AND/(EX)OR relations were omitted from the figure for brevity)

not apply to nodes of the same level, because they have the same level of granularity. Therefore we do not connect nodes of the same hierarchical level.

Using this technique and business knowledge that domain experts possess, we can reason about how an abstract customer need can be specified by more concrete demands, for which a solution (satisfier) can be searched. Figure 5.2 presents a visualization of part of Table 5.1 as a need hierarchy.

Studies we performed in the health sector (Chapter 8), in the energy sector (Chapter 7) and in online news provisioning (de Bruin et al. 2002, Baida, de Bruin & Gordijn 2003), show that the use of above refinement structures requires adding a *context* dimension, since customer needs (or: stakeholder needs, as in de Bruin, van Vliet & Baida (2002)) differ per customer type, and thus the refinement changes per customer type. Different needs, wants, demands and their decompositions may apply to different customer types. In fact, per customer group (or: per stakeholder) we may define a separate need hierarchy. Customer grouping criteria may differ per case. Examples are the nature of consumption (e.g., households vs. industrial customers), the customer's role (e.g., a patient vs. an informal carer of that patient) or the customer's age group (e.g., teenagers typically have a different interpretation of their needs than adults).

For example, the customer want for 'communication' can be refined to three demands: (landline) telephone line, mobile phone line and Internet access. Whereas one customer may require a landline, another may want Internet access and a mobile

phone line, and no landline. This illustrates why supplier stated requirements are not sufficient for e-service bundling: suppliers present services in terms of what they can offer, while customers initially think in vague terms such as 'communication'.

Hence, by following AND/(EX)OR relations we can derive, in collaboration with customers, a set of customer demands based on higher level needs. As described in Figure 5.1, customer demands are a suitable starting point for the *serviguration* process. The next step is to define how potential service outcomes ('resources') satisfy customer demands. This can be done on the demand level, rather than on the more abstract want or need levels.

5.2 Demands are Satisfied by a Service that Provides Certain Resources

The purpose of building a need hierarchy is twofold. First the hierarchy is used to find context depending demands, based on more abstract wants and needs. Second, concrete demands are used to search for services that provide satisfiers (service outcomes, resources) for these demands and for more abstract needs. We employ Feature-Solution graphs (de Bruin & van Vliet 2002, de Bruin et al. 2002) to relate demands and resources. As we explained in Chapter 3, resources are service descriptors; every service requires some resources as inputs, and results in the availability of outcome resources. Configuring service bundles is then the task of designing service bundles that provide a set of desired resources.

A transformation between customer demands (the satisfaction of which is the goal of the service offering) and resources (descriptors of available services, or solutions) can be viewed as a production system consisting of *production rules*, a knowledge representation formalism used in the AI field. Production rules have the form: if situation X is encountered then select solution Y. De Bruin et al. suggested the use of context-aware *Feature-Solution graphs* (FS-graphs) to model these production rules (de Bruin & van Vliet 2002, de Bruin et al. 2002).

FS-graphs capture and document context-sensitive domain knowledge, so that it becomes possible to reason about feasible solutions and the requirements they support. An FS-graph includes three spaces, organized in hierarchies of AND/(EX)OR decompositions:

1. **Feature space**: describes the desired properties of the system (or: service) as expressed by the user. In our case, these are customer demands.

2. **Solution space**: contains the internal system (services) decomposition into resources that are required for or produced by available services.

3. **Context space**: contextual domain knowledge that influences relations between elements of the feature space and elements of the solution space (e.g., customer type, geographic restrictions).

Four types of production rules are used to relate elements of the Feature space (demands) to elements of the Solution space (resources). These include two strong production rules and two weak production rules:

1. *Selection*: if demand D1 exists, resource R1 must be provided by any solution bundle (further referred to as SEL(D1, R1)).
2. *Rejection*: if demand D1 exists, resource R1 must not be provided by a solution bundle (further referred to as REJ(D1, R1)).
3. *Positively influenced by*: resource R1 has a positive influence on satisfying demand D1, but the demand can also be satisfied without the availability of resource R1 (further referred to as POS(D1, R1)).
4. *Negatively influenced by*: resource R1 has a negative influence on satisfying demand D1, but the demand can still be satisfied (although not optimally) when resource R1 is available (further referred to as NEG(D1, R1)).

We refer to the production rules *selection* and *rejection* as strong relations because they allow no flexibility. We refer to the production rules *negatively influenced by* and *positively influenced by* as weak relations because they represent a choice.

The FS-graph offers levels of flexibility as a result of the different decomposition possibilities of features and solutions. An example FS-graph can be found in Figure 5.3. As can be seen, contextual information can change the behavior of a production rule, as modeled by a *context switch*. If a specific context is valid (for a given customer), the related switch node is closed and establishes context-dependent relations between features and solutions (de Bruin et al. 2002). A different context would close a different switch node related to the same demand, resulting in different relations for the same demand. Example contexts are location and customer type (we discuss the topic *context* in detail in Section 5.4).

Our experience in using FS-graphs with domain experts shows that graphs are a good means to visually *communicate* ideas, but when a substantial number of production rules is involved, and in the absence of a software tool to support this task, the use of *Excel* sheets is preferred by domain experts, because the graph becomes too complex to comprehend and to manage. Yet *Excel* also presents a difficulty: it is two dimensional, while the FS-graph is three dimensional. To provide automated support for modeling production rules, constructs of the FS-graph need to be added to the earlier presented service ontology. Figure 3.14 in Chapter 3 shows how we incorporate FS-graph structures in the service ontology.

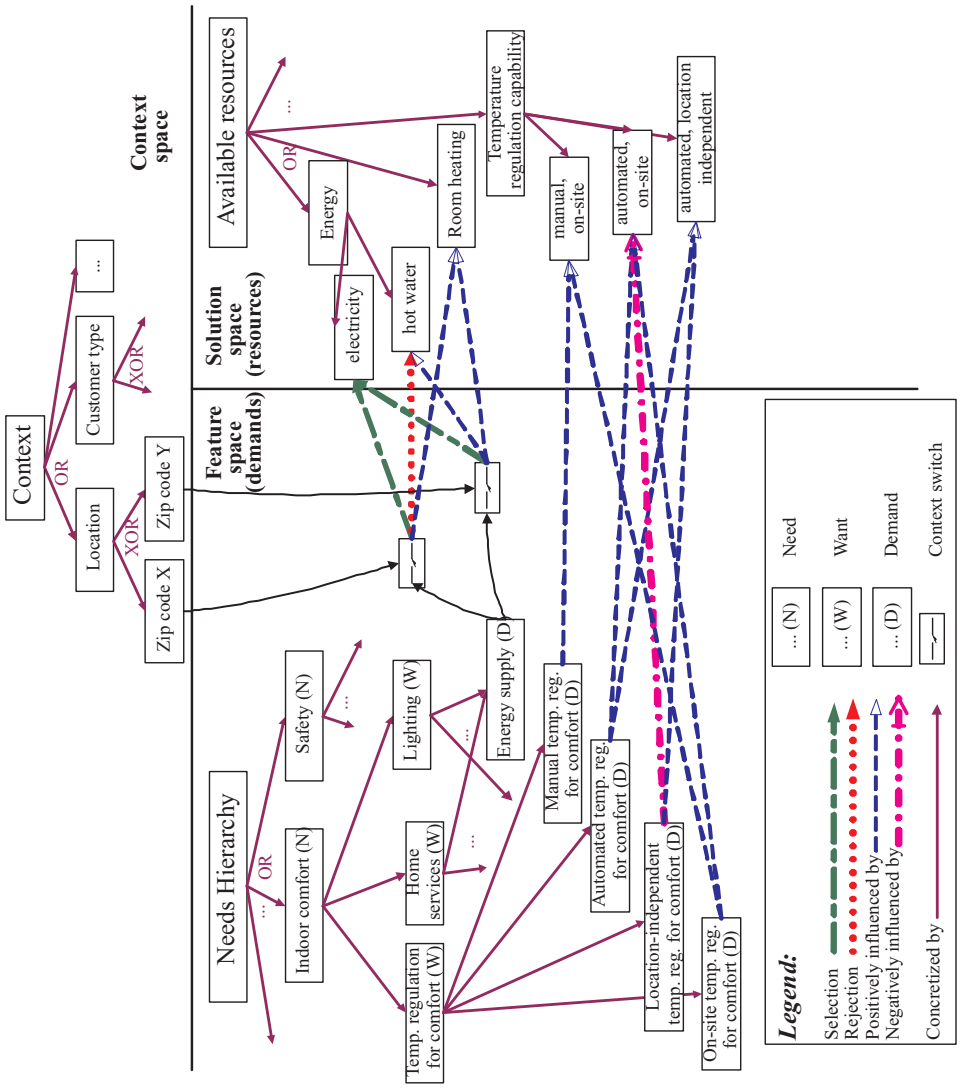

Figure 5.3: Partial FS-graph of the energy study that we discuss in Chapter 7. For visualization reasons we present only a fraction of the need hierarchy graph, and we mention the type of hierarchy (AND/OR/EXOR) explicitly only in a few of the places.

5.3 Reasoning with Production Rules

Very often demands and resources include qualitative and/or quantitative descriptors (referred to as service properties in the service ontology). For instance, in Table 5.1 and in Figure 5.3 we can find the demand for temperature regulation, specified by the descriptors 'manual', 'automated', 'on-site' and 'location-independent'. Service properties may influence production rules. For example, imagine a demand for 'email facilities' that may be specified by the service property 'capacity: small enterprise', and an 'Internet connectivity capability' resource that may be specified by the service property 'connection type: ISDN'. We model two production rules between these demand and resource:

1. SEL('email facilities', 'Internet connectivity capability'): if a customer has a demand for 'email facilities', any solution bundle must include a service that provides an 'Internet connectivity capability' resource.

2. NEG('email facilities' with property 'capacity: small enterprise', 'Internet connectivity capability' with property 'connection type: ISDN'): the availability of a service that provides the resource 'Internet connectivity capability' with property 'connection type: ISDN' in a bundle has a negative influence on satisfying the customer demand for 'email facilities' for a small enterprise.

Two different production rules apply to these demand and resource, depending on the question whether or not the demand and resource are described by service properties. If a customer asks for 'email facilities' for a small enterprise, we search a service that provides an 'Internet connectivity capability' resource without service property 'connection type: ISDN'. This example shows that it does not suffice to model one production rule between any pair (demand, resource). Service properties that describe demands and resources need to be taken into consideration as well.

We assume that a library of service components is known at the start of the *servigura-tion* process, as is required by the definition of "component", presented in Chapter 4. Consequently, we also have a priori knowledge of all resources in our Universe of Discourse, namely all the resources that are related to the services in the service library. Similarly, all demands are defined in need hierarchies. Thus we have a priori knowledge of finite sets of demands and resources, for which production rules need to be defined. Production rules must also be defined a priori, and not derived at runtime, because they represent knowledge which domain experts possess and which cannot be derived from other knowledge involved in *servigurat*ion.

A demand or resource may be specified by more than one service property, and every service property may have a number of valid values. Take for example a demand for energy consumption regulation for budget control (customers wish to regulate their

energy consumption in order to reduce costs). Different customers may have the same demand, but require that energy consumption regulation is done (1) manually or (2) automated. In other words, the demand (say, demand D1) will be specified three times: either with no quality descriptor, or with each of the quality descriptors "manual" and "automated". Consequently, our model will include three demands, using the following unique identifiers:

1. A demand for energy consumption regulation, without specifying any service property (D1)
2. A demand for energy consumption regulation, with property "mode of operation: manual" (D1Q1)
3. A demand for energy consumption regulation, with property "mode of operation: automated" (D1Q2)

The same can be done on the resource side. So we may have a capability resource "temperature regulation capability" (say, R1) with various properties, e.g., "via remote control" and "automated". Our model will then include three resources:

1. A temperature regulation capability resource, without specifying any service property (R1)
2. The same resource, with property "mode of operation: via remote control" (R1Q3)
3. The same resource, with property "mode of operation: automated" (R1Q4)[1]

Between any pair (demand, resource) there may be one or no production rule. However, as we have seen, demand D1 and resource R1 are actually considered as three different demands and three different resources, resulting in a matrix of nine possible production rules between nine pairs of a demand and a resource (see Table 5.2).

The above discussion can also be extended to demands and resources that are described by more than one service property. For example a capability resource "Internet connectivity" may be described by a service property 'download speed: 8000 Kbit/s', as well as by a service property 'upload speed: 1024 Kbit/s'. A very large number of production rules may have to be modeled, resulting in an extensive modeling effort.

Also in the domain of telecommunication services the problem of explosion of combinations has been studied (Keck & Kuehn 1998), and suggested solutions include tools for context generation and information acquisition. Our experience from large

[1] In the current example Q2 and Q4 are identical, but this need not necessarily be the case.

Table 5.2: Matrix of production rules between demand D1 (possibly specified by service properties Q1 or Q2) and resource R1 (possibly specified by service properties Q3 or Q4)

	R1	*R1Q3*	*R1Q4*
D1	(D1, R1)	(D1, R1Q3)	(D1, R1Q4)
D1Q1	(D1Q1, R1)	(D1Q1, R1Q3)	(D1Q1, R1Q4)
D1Q2	(D1Q2, R1)	(D1Q2, R1Q3)	(D1Q2, R1Q4)

scale studies is that the majority of the combinations (demand D1 with property Qx, resource R1 with property Qy) require no production rule, so the modeling effort is reasonable. Customer demands and available services that we model are described typically on a higher level of abstraction than in the case of (executable) telecommunication services as in Keck & Kuehn (1998). For example, we model demands as '(landline) telephone line' and resources as 'Internet connectivity' with a certain download speed and upload speed, but when these services are made operational, a much richer description of QoS (Quality of Service) and desired/available features is required, resulting in a much larger number of feature combinations to deal with.

A study we carried out in the health sector (described in Chapter 8) yielded a means to decrease this complexity. Demands can be divided into clusters, where a cluster includes all demands that are related to a single need. Because resources are solutions for demands, very often also clusters of resources can be observed, that are related (by production rules) to clusters of demands. An important observation from our study in the health sector is that the vast majority of production rules exist between single clusters of demands and single clusters of resources. Only a small number of production rules exist between the same cluster of demands and other clusters of resources (see Figure 5.4).

An important conclusion from this observation is that most modeling work can be performed by modeling experts with a reasonable effort and time investment. We can divide the space of demands and resources into clusters, identify related clusters of demands and resources, and first focus the modeling effort on production rules between these clusters. The vast majority of production rules will be modeled between pairs of clusters. If we look at the example in Figure 5.4, once we identify the three clusters of demands (A, B, C) and the three clusters of resources (X, Y, Z), the majority of production rules will be between pairs of clusters: clusters A and X, clusters B and Y, and clusters C and Z. Since clusters are sets of related demands and solutions

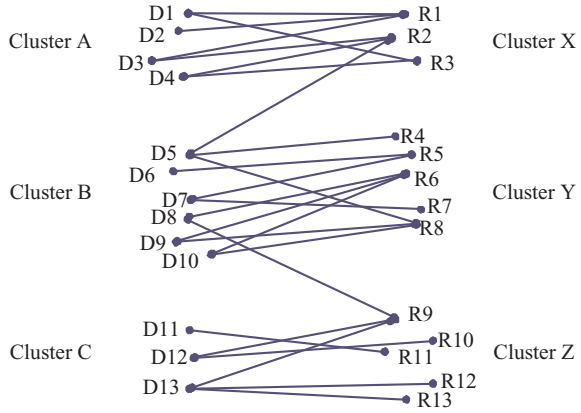

Figure 5.4: Clusters of demands are related to clusters of resources ('D' stands for 'Demand', and 'R' stands for 'Resource')

for these demands, in the health study identifying clusters was natural for domain experts.

Three problems may arise in reasoning with production rules. The first problem occurs when various production rules involve the same resource. This may cause conflicts between production rules. Imagine that we have two demands, D1 and D2, one resource R1, and the following production rules: SEL(D1, R1) (meaning that resource R1 must be selected if demand D1 is triggered) and REJ(D2, R1) (meaning that resource R1 mustn't be selected if demand D2 is triggered). A conflict occurs when a customer has demands D1 *and* D2. On the one hand resource R1 must be part of any service bundle, and on the other hand it may not be part of a solution (bundle). In cases that we modeled in the health sector and in the energy sector, this problem was only theoretical, but it did not appear in practice. Namely, in reality when two conflicting production rules involve two different demands D1 and D2, domain experts declared that these demands cannot co-exist, so the conflicting production rules involving (D1, R1) and (D2, R1) will not be triggered at the same time.

In the second problem conflicts occur when two production rules involve the same demand D1 with different service properties (e.g., D1Q1 and D1Q2) and a single resource R1Q3. Demand D1Q1 may require resource R1Q3, while demand D1Q2 has a rejection relation with R1Q3. What must be done when both D1Q1 and D1Q2 apply? This situation is different from the first problem, because here the conflicting production rules involve the same demand (only with different service properties), while the first problem involved two completely different demands.

The third problem occurs when a demand and a resource have a production rule that applies independently of any service property, as well as production rules that apply

only when specific service properties are specified. Let us take as an example the earlier presented demand 'energy consumption regulation for budget control' (D1) with the service property 'mode of operation: manual' (Q1), and resource 'temperature regulation capability' (R1) with service property 'mode of operation: automated' (Q4). Two production rules are relevant here:

- POS(D1, R1): resource R1 has a positive influence on satisfying demand D1. This is a so-called 'global' production rule: it does not take into consideration the properties of D1 and of R1.
- REJ(D1Q1, R1Q4): when demand D1 is specified by property Q1, the solution must not include a resource R1 with property Q4. This is a so-called 'local' production rule: it holds for D1 and R1, only when they are specified by service properties Q1 and Q4 respectively.

To automate reasoning with production rules, one must know how the two production rules should be used together. Does the POS production rule apply when a user specifies demand D1 with quality descriptor Q1, because it is global (it holds for any pair (D1, R1), independent of their properties), or does the REJ production rule apply because it is a strong relation (while POS is a weak relation) or because it is more specific? Similar conflicts may occur also between other pairs of production rules, as we will show.

In the next sub-section (Section 5.3.1) we discuss the first and the second problems, and in Section 5.3.2 we provide a solution for the third problem.

5.3.1 Conflict Detection and Resolution

Feature-Solution graphs (FS-graphs) use four different types of production rules, as described above. A conflict between production rules is a situation in which any of the following pairs of production rule types is triggered for the same resource: (1) SEL and REJ; (2) SEL and NEG; (3) REJ and POS; (4) NEG and POS.

Conflict management (detection and resolution) may be performed either offline (a priori), online (at runtime) or as a combination. In other words, either it is performed *before* users configure their own service bundle, or it is performed only once users trigger the *serviguration* process, or some part is performed a priori, and another part at runtime. Our discussion hereunder considers mainly conflict detection. We limit our discussion on conflict resolution to deciding whether a conflict can be solved without changing the customer's demands. We do not investigate different ways to alter a set of conflicting demands so that a solution can yet be found.

The service ontology can be used either for a business analysis, or to enable customers design service bundles by themselves through a website. In the first case

(business analysis), there is no explicit need to perform conflict detection and resolution in advance, because the whole process is of an exploratory nature, so conflict management is an integral part of the analysis. This is not the case in the second scenario. When developing a website through which customers can design service bundles that suit their situation, online conflict management is not feasible for the following reasons. First, it isn't realistic to expect customers to solve conflicts in the reasoning engine of an information system that they use. Second, if suppliers let customers influence the reasoning engine of the information system, customers may request service bundles that are financially not interesting for service suppliers. Service suppliers are not interested in offering these service bundles, and would therefore typically want to limit the space of possible solutions (service bundles) that an information system can offer to customers. This requires that customers cannot influence the reasoning engine of the system, and therefore conflict resolution should take place a priori.

Next, we discuss *how* to perform conflict resolution. Baida, de Bruin & Gordijn (2003) used FS-graphs for an assessment of an e-business case study. They handled conflicts based on their type:

- A *major conflict* is a conflict between two strong production rules. It involves a SEL relation and a REJ relation. No solution is possible, so no service bundle can satisfy the given demands.
- A *minor conflict* is a conflict between two weak production rules. It involves a POS relation and a NEG relation. Satisfying the demands is possible, but it requires compromises (typically, the suggested service is not "exactly" what the customer wanted; yet the customer may accept this solution if no better option exists or if its price is significantly lower than the price of other solutions).
- The third type of conflict involves a strong production rule and a weak production rule: either a SEL relation and a NEG relation, or a REJ relation and a POS relation. In these cases we analyze the impact of the conflict, and classify it as a major one or as a minor one. We refer to this as "the third type of conflict".

We modeled production rules in studies in the health sector (Chapter 8) and in the energy sector (Chapter 7), analyzed the nature of conflicts, and tried to apply the above classification of conflicts. We found that in numerous cases reflecting the first problem discussed earlier (a conflict that involves two different demands), the conflict is only theoretical. In practice when two conflicting production rules involve two demands D1 and D2, these demands cannot co-exist, so the production rules involving (D1, R1) and (D2, R1) will not be triggered at the same time. In cases concerning conflicts between two 'local' production rules involving the same demand (but with different service properties):

Table 5.3: Conflict resolution in case of conflicting production rules

Conflict type	*Conflict resolution*
Major conflict	No solution exists (no service bundle can satisfy the demands)
Minor conflict	Domain experts should decide whether the conflict can be solved or not. If the conflict is declared as solvable, the resource at hand may (but need not necessarily) be included in a bundle (i.e., consider only the POS relation; disregard the NEG relation). If the conflict is declared as unsolvable, the resource at hand may not be part of a bundle. Yet, because the resource didn't necessarily have to be part of a bundle (as it did not have the selection relation), a solution is yet possible.
Third type of conflict	Our experience shows that it would be safe to say that no solution exists (no service bundle can satisfy the demands). Yet, domain experts may still wish to analyze every such case independently to see whether there are some exceptional cases where a solution may exist nevertheless.

- No major conflicts were identified
- Minor conflicts turned to be divergent: either the conflict could be ignored (i.e., the POS relation was stronger than the NEG relation), or the conflict was unsolvable (and hence the resource at hand could not be part of a solution).
- In all cases of the third type of conflict, there was no solution for the conflict (and hence no service bundle could satisfy the demands).

Based on these findings and on Baida, de Bruin & Gordijn (2003), we determine rules for conflict resolution. These are described in Table 5.3. As can be seen from the table, mainly minor conflicts require human intervention to understand the nature of the conflict, and to assess how the conflict should be handled.

5.3.2 Granularity in Production Rules

We described the problem that occurs when a demand and a resource have a 'global' production rule (that applies independently of any service property), as well as 'local'

production rules (that apply only when specific service properties are specified). In the current section we analyze this problem.

The need for 'local' production rules next to 'global' production rules stems from differing levels of reasoning. In reality, most demands and resources are specified by some service properties. Various similar demands and resources may exist, that differ only in some property, or in the value of a property. For example, two Internet connectivity capability resources may exist, both with the service property 'download speed' and 'upload speed', but yet every resource will have different values for these properties (i.e., different download/upload speed). Reasoning with 'global' production rules, we may say that a demand for email facilities may be satisfied by an Internet connectivity capability resource without specifying any properties (i.e., without requiring certain download speed or upload speed). This is a 'global' production rule. However, if the same demand is set with a quality descriptor 'capacity: household', we will set requirements also on the download/upload speed, resulting in 'local' production rules. Note that the service ontology allows us to describe whether the values of resources in production rules specify a minimum value, a maximum value or an exact value. A production rule may then define that when a demand for 'email facilities' is set with the service property 'capacity: household', a resource 'Internet connectivity capability' must be selected with service property 'download speed' with a value of *at least* 800 Kbit/s. This is facilitated by the concept 'requirement expression' in the ontology.

Imagine we have a 'global' production rule between demand D1 and resource R1 (without specifying service properties), as well as a 'local' production rule between demand D1 with property Q1 (further referred to as D1Q1) and resource R1 with property Q2 (further referred to as R1Q2, it does not matter whether Q1 and Q2 are the same property or not). We need to define the relation between the 'global' production rule between D1 and R1 ('parents') and the 'local' production rule between D1Q1 and R1Q2 ('siblings'). In our discussion we use the following terminology:

- When we say "parents" we refer to a pair where both the demand and the resource are not specified by service properties, i.e., (D1, R1).
- When we say "siblings" we refer to a pair where either the demand, or the resource, or both are specified by service properties, i.e., (D1, R1Q2), (D1Q1, R1) or (D1Q1, R1Q2). These three pairs are the siblings of (D1, R1). This can be generalized to a situation where the demand and/or resource are specified by multiple service properties. In this case, also (D1, R1Q2Q3), (D1Q4, R1Q2Q3) and so forth are siblings of (D1, R1).

We further use the following guidelines:

- A 'global' production rule between demand D1 and resource R1 applies whenever demand D1 is set and is not specified by quality descriptors (service properties). It holds for resource R1, disregarding its quality descriptors.
- A 'local' production rule between D1 and R1Q2 restricts R1 only when R1 is specified by a property Q2, whenever demand D1 is set without being specified by any service properties.
- A 'local' production rule between D1Q1 and R1Q2 restricts R1 only when it is specified by a service property Q2, whenever demand D1 is set and is being specified by service property Q1.

As a general approach, we say that a production rule of siblings is more specific than the production rule of the parents, and therefore it overrides the parents' production rule. Yet, the relation between the parents' production rule and the siblings' production rule depends on the types of production rule that the parents and the siblings have. Thus, every pair of production rule types (production rule of parents, production rule of siblings) is a separate case. In the rest of this section we discuss every possible case separately. Throughout our discussion we use the same notation as presented above, where Rx and Dy denote a resource/demand (independent of their service properties), and RxQa and DyQb denote a resource/demand that is specified by a service property Qa or Qb respectively. We provide real-life examples for various cases throughout the discussion.

Cases where weak relations are involved (POS or NEG) can be used to define an ordering among solutions, as we show in Figure 5.5. For example, a solution that involves a POS relation is better than a solution that does not involve such a relation; a solution that involves a NEG relation is worse than a solution that does not involve such a relation. Our experience in modeling real-world situations shows that when a NEG relation is involved, and there exist solution bundles that do not involve this relation (i.e., solutions that do not provide a resource as specified by a NEG relation), in fact there is no need to offer those bundles that provide the resource for which a NEG relation exist, because customers would not choose for it (in Figure 5.5, small enterprises seeking for 'email facilities' will not select service ISP(3), when the other options exist). Therefore, if there are solutions that do not provide a resource specified by a NEG relation, we do not generate solutions that *do* include this relation (in Figure 5.5 this means that service ISP(3) will not be offered as a solution).

Case 1: parents (D1, R1) have a SEL relation
Example:
Demand D1: email facilities
Resource R1: Internet connectivity capability resource
Production rule: SEL(D1, R1)
If the parents have a SEL relation, a service that provides resource R1 *must* be part

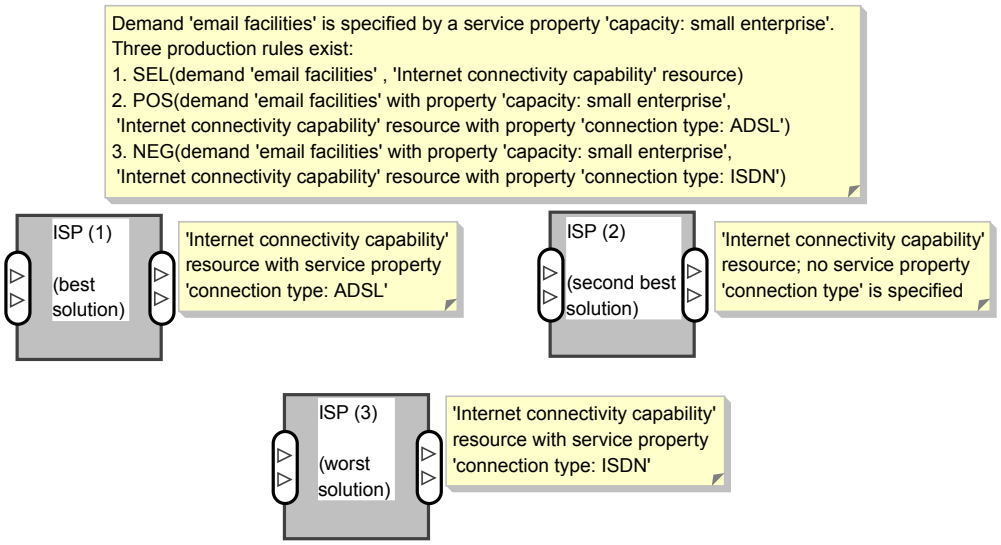

Figure 5.5: Ordering among solutions. In the current example there exist three different ISP services that provide an 'Internet connectivity capability' resource. Based on the first production rule, a service that provides this resource must be part of every solution. The POS and NEG production rules imply an ordering among solutions. Service ISP(1) is a better solution than service ISP(2) due to the POS production rule. Service ISP(3) is a worse solution than service ISP(2) due to the NEG production rule.

of any service bundle. Let us examine how the siblings' relations can be taken into consideration.

Case 1a: siblings have a SEL relation
Example:
Demand D1 is specified by service property Q1 'capacity: household'
Resource R1 is specified by service property Q2 'download speed: 800 Kbit/s'
Production rule: SEL(D1Q1, MINIMUM R1Q2)
The siblings' SEL relation is more specific than the parents' SEL relation. If the siblings' SEL relation specifies properties for R1, for example R1Q2, any solution must include R1 with these properties.

Case 1b: siblings have a REJ relation
Example:
Demand D1 is specified by service property Q1 'capacity: medium enterprise'
Resource R1 is specified by service property Q2 'connection type: ISDN'
Production rule: REJ(D1Q1, R1Q2)
Due to the parents' relation, any solution must include R1. However, the siblings'

REJ relation disqualifies any resource R1 that is specified by the property Q2 'connection type: ISDN'. A solution service bundle must then include an instance of R1 that is not specified by Q2. If no service can provide R1 with other properties than Q2, there is no solution (i.e., no service bundle can satisfy the given demand).

Case 1c: siblings have a POS relation
Example:
Demand D1 is specified by service property Q1 'capacity: household'
Resource R1 is specified by service property Q2 'connection type: ADSL'
Production rule: POS(D1Q1, R1Q2)
The SEL relation of the parents means that a service that provides resource R1 must be part of any service bundle. There may be many instances of this resource, and many services that provide this resource, with different service properties. We distinguish between two situations:

- If the siblings have a POS(D1Q1, R1) relation (i.e., when the resource is not specified by service properties), the parents' relation is stronger, so the sibling's relation should be ignored.
- If the siblings have a POS(D1Q1, R1Q2) or POS(D1, R1Q2) relation (i.e., when the resource is specified by service properties), those services that provide R1 with property Q2 are better solutions than services that provide R1 without Q2, because they comply with the 'global' (more general) production rule as well as with the 'local' (more specific) production rule. We exemplify this situation in Figure 5.5. This reasoning can be used to create an ordering of solution bundles (i.e., as an optimality criterion).

Case 1d: siblings have a NEG relation
Example:
Demand D1 is specified by service property Q1 'capacity: small enterprise'
Resource R1 is specified by service property Q2 'connection type: ISDN'
Production rule: NEG(D1Q1, R1Q2)
The natural way to interpret the combination of the parents' SEL relation and the siblings' NEG relation is: "a service that provides resource R1 must be part of any service bundle; a service that provides resource R1 without property Q2 is a better solution than a service that provides resource R1 with property Q2". However, as discussed in Section 5.3.1, we refer to a situation where the relations SEL and NEG co-exist as "conflict of the third type". As we showed in Table 5.3, in such conflicts there is no solution bundle that can satisfy the given demand. Hence, case 1d should be understood as "a service that provides resource R1 *without* property Q2 must be part of any service bundle" (the NEG relation behaves in fact as a REJ relation). In our example, any solution bundle has to include a service that provides an 'Internet connectivity capability' resource, and this resource may not be specified by

the service property 'connection type: ISDN'. If all available 'Internet connectivity capability' resources are specified by this property, no solution exists.

Case 2: parents (D1, R1) have a REJ relation
If the parents have a REJ relation, a service that provides resource R1 *mustn't* be part of a service bundle. In this case there is no logic behind modeling any other relation (SEL, POS, NEG) on the siblings level. Modeling a REJ relation on the siblings level is possible but redundant, so it can be ignored.

Case 3: parents (D1, R1) have a POS relation
Example:
Demand D1: computer gaming
Resource R1: Internet connectivity capability resource
Production rule: POS(D1, R1)
If the parents have a POS relation, a service that provides resource R1 may (but need not necessarily) be part of a service bundle. For example, an Internet connection may have a positive influence on satisfying a customer demand for computer gaming, as many computer games take place over the Internet. Yet, the demand may be satisfied also without an Internet connectivity, for example by computer games for a stand-alone PC. Let us examine how the siblings' relations can be taken into consideration.

Case 3a: siblings have a SEL relation
Example:
Demand D1 is specified by service property Q1 'number of players: unlimited'
Resource R1 is not specified by any service property
Production rule: SEL(D1Q1, R1)
Computer games where the number of potential players is said to be unlimited are Internet-based games, and therefore an Internet connectivity capability resource is *required* (the SEL relation is strong: any bundle must include a resource R1 as specified by the siblings' relation). Yet, this situation is quite theoretic, as we did not encounter it in our studies in complex domains.

Case 3b: siblings have a REJ relation
Example:
Demand D1 is not specified by any service property
Resource R1 is specified by service property Q2 'connection type: ISDN'
Production rule: REJ(D1, R1Q2)
The REJ relation is stronger than the POS relation. A solution service bundle must not include a service that provides the resource as specified by the sibling's relation. In our example, an ISDN connection is not enough to support online computer gaming, and therefore R1 with property Q2 is not a suitable solution.

Case 3c: siblings have a POS relation
Example:
Demand D1 is not specified by any service property

Resource R1 is specified by service property Q2 'connection type: ADSL'
Production rule: POS(D1, R1Q2)
The parents' weak POS relation means that also bundles may be generated that do not include a service that provides R1. For all the bundles that *do* include R1:

- If the siblings have a POS(D1Q1, R1) relation (i.e., when the resource is not specified by service properties), this relation does not add any knowledge, and can be ignored.
- If the siblings have a POS(D1Q1, R1Q2) or POS(D1, R1Q2) relation (i.e., when the resource is specified by some service properties), services that provide R1 with property Q2 are better solutions than services that provide R1 without Q2 because they comply not only with the 'global' production rule, but also with the more specific, 'local' one. This reasoning can be used to create an ordering of solutions bundles (i.e., as an optimality criterion). It is similar to the reasoning presented in Figure 5.5, with one difference: the parents have a POS production rule instead of a SEL production rule.

In our example, a solution that offers Internet connectivity with the property 'connection type: ADSL' would be a better solution than (1) a solution that does not offer Internet connectivity at all, and (2) a solution that offers Internet connectivity without knowledge of the connection type.

Case 3d: siblings have a NEG relation
In case the siblings' relation specifies R1 with some service property Q1, the siblings relation overrides the parents relation. We prefer that solutions include no resources with NEG relations, but if there are no solutions that do not include R1Q2, we prefer providing R1Q2 to having no solution. Once again, this reasoning can be used as an optimality criterion. Also this case is similar to the one in Figure 5.5: only if services ISP(1) and ISP(2) do not exist, will we offer service ISP(3) as a solution.

Case 4: parents (D1, R1) have a NEG relation
Our experience in modeling real-world cases showed that the NEG relation appears on the siblings level, and not on the parents level. Therefore, the analysis of this case remains theoretical. If the parents have a NEG relation, a bundle that does not include a service that provides resource R1 can satisfy the given customer demand better (i.e., is a better solution) than a bundle that includes a service that provides resource R1. Consequently, if there are solutions (bundles) that do not include R1, these are preferred. Once again, this can be used as an optimality criterion. Let us examine how the siblings' relations can be taken into consideration.

Case 4a: siblings have a SEL relation
The siblings' relation is more specific than the parents' one, so it overrides the parents' relation.

Case 4b: siblings have a REJ relation
The REJ relation is stronger than the NEG relation. A solution service bundle must not include a service that provides the resource as specified by the siblings' relation.

Case 4c: siblings have a POS relation
In the case of a POS(D1, R1Q2) relation or a POS(D1Q1, R1Q2) relation (i.e., when the resource is specified by some service properties), the siblings relation overrides the parents relation, because it is more specific. As a result, a service that provides resource R1Q2 may add value to (but it is not required to be present in) a service bundle. Solutions that include R1Q2 are therefore superior to solutions that do not provide R1Q2 (to be used for ordering solutions).

Case 4d: siblings have a NEG relation
First we apply the parents' relation, meaning that we do not generate bundles that include R1, unless there is no solution without R1. Only in the latter case, do we consider the siblings' NEG relation. When only solutions exist that include resource R1, and the siblings have a NEG relation as well, a bundle that provides resource R1 with properties as specified by the siblings' NEG relation can be considered as a candidate solution. Yet, as it involves the NEG relation on the parents level and on the siblings level, it is likely to be a bad solution. We did not encounter such cases in our studies.

5.4 Context: How One Customer Differs from Another

The service value perspective of our service ontology – including the concepts *needs, wants, demands, sacrifice* and *service property* – reflects a customer view on services. As such it is by definition context-sensitive: every customer or customer type may have a different viewpoint on a service, based on his/her situation (time, location, role), on different expectations and on past experiences (Zeithaml et al. 1990). In this section we show how the context of a customer can be taken into consideration in the design of customer-tailored service bundles.

A customer's context may either relate to his personal situation or to his belonging to a target group. For example, when we offer medical services to patients, we take into consideration their *personal* medical dossiers, with knowledge about their health state. On the other hand, when we offer services to customers without knowledge of them as individuals, we base our offering on more general customer characteristics, e.g., the customer's age group. Customers who share similar needs/demands in similar contexts (e.g., the demand for energy supply for industrial customers within a geographic region) are said to belong to the same *market segment* (Kotler 1988): "a market segment is defined as a concept that breaks a market, consisting of actors, into segments that share common properties".

We model this information in the service ontology using the concept *context*, reflecting the physical and social situation of (in our case) customers of services that we model. The concept 'context' in the service ontology has the attributes name and value, for example 'name: age, value: 65' or 'name: customer type, value: informal carer'. Multiple contexts may be valid simultaneously.

As we will show in Chapter 8, two customers may have the same demand, and yet require different services to satisfy this demand because of their different ages. Hence, the transformation (captured by production rules) between needs and available resources (that are provided by services), and the choice of services to be included in a bundle, depend on the context of a given customer, or a customer group. Service bundles are to be designed for customers who have certain needs, and are in a certain context.

Throughout this thesis we refer to Figure 5.1, presenting a simplification of the whole *serviguration* process. Context information is taken into consideration explicitly twice in the process:

1. Some context information describes the conditions under which a benefit (resource) can satisfy a demand (we refer to this as "context on the resource level"). We model such relations by defining that production rules may depend on a customer's context.
   Example:
   Demand D1: Discussion group concerning how to cope with the changing behavior of a patient
   Resource R1: Coping advice for informal carers
   Production rule: SEL(D1, R1)
   Context: Customer type: informal carer
   Explanation: The SEL relation will be triggered only in queries where the customer type is 'informal carer'. Consequently, when an informal carer asks for 'discussion group concerning how to cope with the changing behavior of a patient', we will search for a service that offers 'coping advice for informal carers'. When a different customer (e.g., a patient) has the same demand, the SEL production rule will not be triggered, and therefore we will not offer a service that provides coping advice for informal carers. Different resources exist for different customer types because patients and informal carers require different advice and support (yet, a single service may provide resources for both).

2. Some context information describes the conditions under which a whole service element qualifies (or does not qualify) as a solution (we refer to this as "context on the service level"). This is supported by the relation "service element has context" in the service offering perspective. Services that require

another context than the one specified by the current customer are not valid candidates to be included in a service bundle by the service configuration task.
Example:
A service for renting appliances (e.g., a wheelchair) is provided only to customers who live in a certain region. We model this geographical restriction as context information, related to the appliances rental service.

Some context information can be considered as global assumptions that narrow the scope of the information we model and of information systems that can use this model. For example, when developing an information system for service offerings within a specific geographic region, the location is assumed to be a global assumption, and it is not necessary to explicitly constrain all service offerings to that region. Global assumptions of a model (of services and customer needs) are considered to be known by all the users of the model, and are not made explicit in the *serviguration* process. Examples of context information are given in Chapter 8, where we describe a study in the health sector.

5.5 Summary and Discussion

The research questions in this thesis indicate that we develop a model to facilitate software-aided service bundling, based on supply-side and demand-side business logic. In Chapter 4 we showed how to incorporate supply-side business logic into service bundling, or configuration. The actual configuration of services into service bundles, as discussed in the previous chapter, assumes that bundling requirements are described in terms of the actual offerings: resources that services provide. However, because services are satisfiers of customer needs, bundling requirements should be derived from an understanding of these needs. In real-world this process is typically performed by service personnel. Finding service offerings by computer-supported customer-need reasoning requires that this reasoning process is modeled, and that knowledge is represented in a computer-interpretable way. In this chapter we showed how an understanding of customer needs serves us to derive requirements for the actual service bundling/configuration task. We include concepts for this reasoning process in our service ontology, and employ reasoning techniques from the field of computer science and artificial intelligence to perform this customer-need reasoning.

First, we describe customer needs in hierarchies including three levels: needs, wants and demands, and use AND/(EX)OR decompositions to reason about how concrete demands fulfill higher-level needs. Needs are often too vague to find an actual solution. Therefore we derive a set of customer demands that can fulfill a need.

Second, once we have derived a set of desired customer demands, we use production rules to reason about the suitability of services as satisfiers of these demands. Every

service provides certain benefits, referred to as resources. Multiple services typically offer the same or similar resources. Having derived a set of desired customer demands, we use production rules to derive a set of resources (service benefits) that can satisfy customers. This set serves as input for the actual service configuration process, in which bundles are designed that include services that provide the desired resources.

The reasoning capacity that we present in this chapter represents a high-level understanding of customer behavior, which is in line with the generic nature of the service ontology. A coarser understanding of customer behavior, as studied in marketing research, will help find more accurate solutions to customer demands. However, because a coarse understanding of customer behavior is mostly limited to a certain industry, product type, marketing channel, country or customer type, it would require compromising on the generic nature of our ontology.

We considered in this chapter the complexity problem, caused by the large number of pairs (demand, resource) for which domain experts have to consider whether a production rule must be modeled. We observed that clusters of demands and resources can be identified, such that the vast majority of production rules will be between pairs of clusters. Only limited effort needs to be put into modeling production rules outside these pairs of clusters, so the modeling effort is reasonable. At the same time we acknowledge that more empirical studies are required to make a sound statement about the complexity problem in modeling production rules.

Also, we investigated production rules involving only one demand and one resource. These were enough to model realistic and complex domains. Yet, more empirical studies are required to investigate the necessity for production rules involving multiple resources (e.g., IF demand X THEN resource Y *or* Z).

An interesting observation is that we perform conflict resolution in the relations (production rules) between features (demands) and solutions (resources). This is opposed to conflict resolution in software engineering (van Lamsweerde et al. 1998) and in software architecture, where conflicts are managed on the feature side: goals and requirements. A possible explanation for this difference is the fact that in software design all requirements and goals refer to the same single artifact: the system to be developed. In the case of service bundling, on the other hand, customer demands need not depend on each other, and the solution may comprise of totally independent services (artifacts) that can be consumed at different times. For example, a customer may have a demand for home entertainment as well as entertainment outside home. These two demands do not conflict, because a solution service bundle may include a service that delivers home entertainment (e.g., a TV subscription) and a service that delivers entertainment outside home (e.g., a subscription for the National Ballet), and the two may be consumed independently, at different times and locations.

Chapter 6

Tool Support

***Note**: In this chapter we discuss the implementation of our service ontology in software tools. Parts of this chapter have been discussed in Baida, Gordijn, Morch, Sæle & Akkermans (2004) (the 17th Bled eCommerce Conference, Bled 2004), in Akkermans, Baida, Gordijn, Peña, Altuna & Laresgoiti (2004) (an article in the IEEE Intelligent Systems magazine), in Baida, Gordijn, Sæle, Akkermans & Morch (2005) (the 17th International Conference on Advanced Information Systems Engineering, CAiSE 2005) and in Dröes, Meiland, Doruff, Varodi, Akkermans, Baida, Faber, Haaker, Moelaert, Kartseva & Tan (2005) (an article in Medical and Care Compunetics 2).*

In the framework of our research we implement ontologies in software tools for three goals. First, tools are used for validating the computational adequacy of the underlying ontology. Second, they help validate the theoretical adequacy of the underlying ontology. Third, they serve as a means to communicate with domain experts, who need not be aware of the technical intricacies of an ontology, and work better with a graphical user interface. In fact, the need for a graphical representation was a requirement for our ontology, as discussed in Section 2.4.4. Other goals exist in other contexts, mainly ontology-based software tools are used for application integration in semantic web initiatives.

In Section 6.1 we report about implementing the service offering (supplier) perspective of the service ontology in software tools. Section 6.2 presents a software tool to support the service value (customer) perspective as well as the supplier perspective, with an emphasis on the customer perspective. Finally, in Section 6.3 we provide concluding remarks.

6.1 Tool Support for the Supplier Perspective

Within the EC-funded IST project OBELIX we developed a configuration tool and a service modeling tool. The tools have been integrated to configure service bundles. The service modeling tool is part of the research effort that lead to this thesis, and is based on our *serviguration* service ontology; the configuration tool is part of research done by the Spanish project partner *Fundación LABEIN*, and is based on the generic configuration ontology we discussed in Chapter 4.

Figure 6.1 shows how these applications collaborate in service configuration:

1. First we model services, business rules and bundling requirements using the service modeling tool.
2. Next, we use a transformation module between the two tools to transform service ontology terminology onto configuration ontology terminology.
3. Next, the configuration tool computes all *suitable configurations*.
4. Suitable configurations are then transformed back to service ontology terminology, and fed back to the service modeling tool.
5. Suitable solutions are visualized in the service modeling tool, and presented to users.

All communication between the service modeling tool, the configuration tool and the transformation module takes place by exchanging RDF files with instantiations of the service ontology and the configuration ontology.

6.1.1 Configuration Tool

Our Spanish project partner *Fundación LABEIN* developed a generic configuration tool within the EC-funded IST project OBELIX. The configuration tool uses the previously presented configuration ontology to support collaborative configuration of goods and services. As explained in Altuna et al. (2004), the tool can import and export RDF ontology files, and provides four different interfaces, through which other applications can use the configuration tool as a module:

1. A *problem specification interface* to define configuration requirements.
2. A *constraint specification interface* to specify domain constraints. This interface is facilitated by a graphical user interface.
3. A *configuration execution interface* to carry out a configuration task.

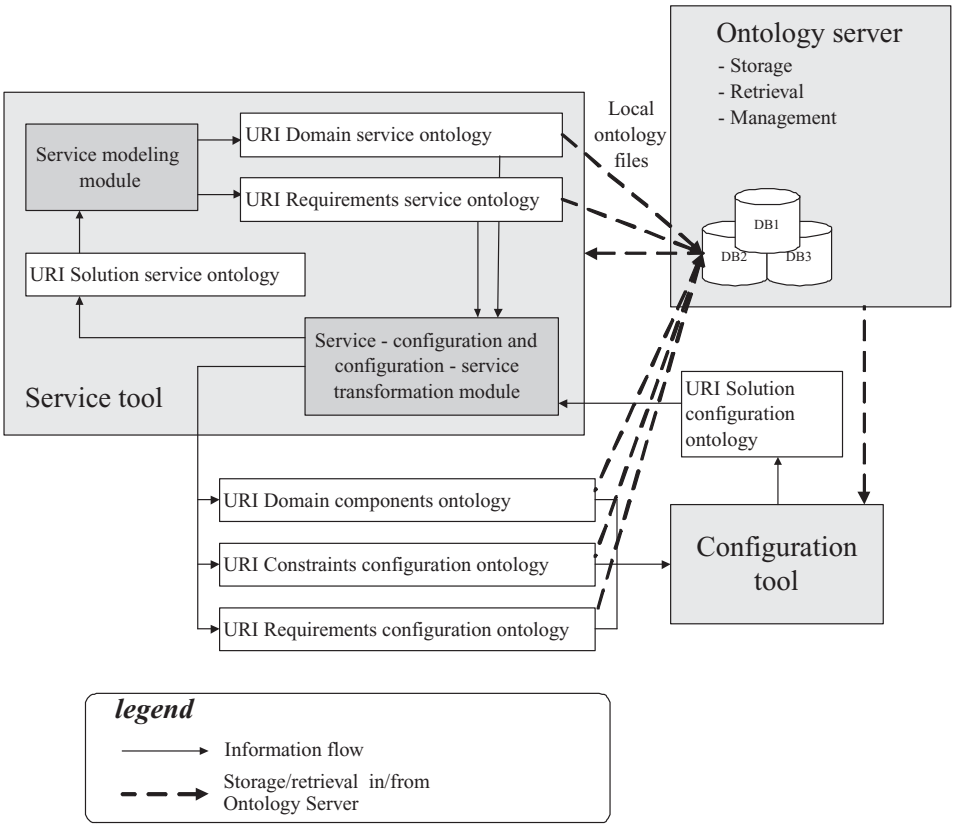

Figure 6.1: How ontology-based tools collaborate in service configuration. Arrows indicate the flow of RDF files; broken arrows indicate a possibility to store/retrieve RDF files. Source: Akkermans, Baida, Gordijn, Peña, Altuna & Laresgoiti (2004).

4. A *configuration solution interface* to retrieve the solutions of a configuration task by the domain application.

The tool architecture is further described in Altuna et al. (2004), and is beyond the scope of this thesis.

6.1.2 Service Tool

The service tool we developed helps reduce the complexity of designing service bundles that involve a joint offering of a number of enterprises. To this end, a software tool is required that enables domain experts, business developers and consultants to model business logic and services from a business value perspective, so that service bundles may be generated by the software.

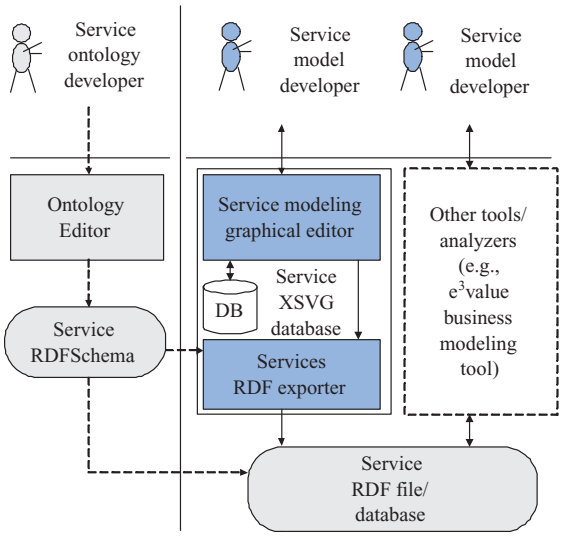

Figure 6.2: Service modeling tool architecture overview. Source: Akkermans, Baida, Gordijn, Peña, Altuna & Laresgoiti (2004).

Tool Functionality and Architecture

The use of software for service bundling necessarily means that business knowledge on services needs to be represented (modeled) formally. Modeling services formally becomes very complex when we deal with a non-trivial case involving a multitude of services, offered by a variety of suppliers. Our experience shows that performing such a task with a generic ontology editor as *Protégé* or *OntoEdit* is a very time-consuming task, and mistakes are very likely to happen. Nevertheless the modeling effort must take place if suppliers wish to use the service ontology for a business analysis, or to offer their customers the possibility to configure service bundles on-line. To solve this complexity problem we implemented a prototype service modeling tool within the EC-funded IST project OBELIX. By providing an easy to use graphical user interface, the tool hides the underlying service ontology from its users. Once a user graphically models services and business rules based on the service offering perspective of the service ontology, the tool stores the model in xsvg format, which can later be re-read and visualized. The tool can also generate ontology-based RDF representations of the visualizations. These are not readable for an average user of the tool, but they provide the formalism that software requires to perform software-aided service bundling. The architecture of this tool is depicted in Figure 6.2.

Next, we designed and developed in collaboration with *Fundación LABEIN* a transformation module that transforms RDF files representing knowledge in terms of the service ontology to knowledge in terms of the configuration ontology, and vice versa.

This transformation module has been embedded in the configuration tool of *Fundación LABEIN* (see Section 6.1.1).

As a result, the service tool can use the configuration tool as a module for configuring service bundles. First a library of services can be modeled using the service tool. Next, a graphical user interface enables the user of the service tool to define requirements for a desired service bundle. Next, the user can trigger a service bundling (configuration) task. The transformation module creates service ontology based RDF files to represent domain knowledge and bundling requirements, based on the visual model created by users. These files are then transformed into configuration ontology based RDF files, which are fed into the configuration tool as input for the configuration task. The resulting solutions are given in configuration ontology based RDF files. The transformation module transforms the solutions into service ontology based RDF files. These are fed back into the service tool, where a visualization module transforms the machine-readable representation of service bundles into a visual representation. All RDF files are stored locally. The whole process is described in Figure 6.1.

Tool Usage

Support for modeling domain knowledge and configuring service bundles are the two main functionalities that the OBELIX service tool offers. Other functionalities, such as RDF(S) generation and visualizing RDF representations of the service ontology support the two main functionalities. In this section we provide a short description of how the tool can be used; this is not intended as a user manual.

Services, visualized as shown in Chapter 3, can be modeled through a drag-and-drop user interface (see Figure 6.3). Every service has an input interface on its left side, and an outcome interface on its right side. A right-click menu provides the possibility to model resources, which can be specified by user-defined qualitative and quantitative service properties in a new screen (see Figure 6.4). Once a resource has been modeled, it can be assigned to a service port of a service element by clicking on that port, implying that this resource is required by the relevant service as input (if the service port belongs to an input interface) or becomes available when consuming this service (if the service port belongs to an outcome interface). Next, dependencies between services (in fact, constraints for the configuration process) can be modeled by selecting labels on bi-directional arrows between service elements (see Figure 6.5).

A right-click menu provides the possibility to define bundling requirements for the configuration process. Bundling requirements are a set of service outcomes and possibly service inputs (resources) that a desired service bundle must include. Users can determine the desired values of service properties of resources that they require, and

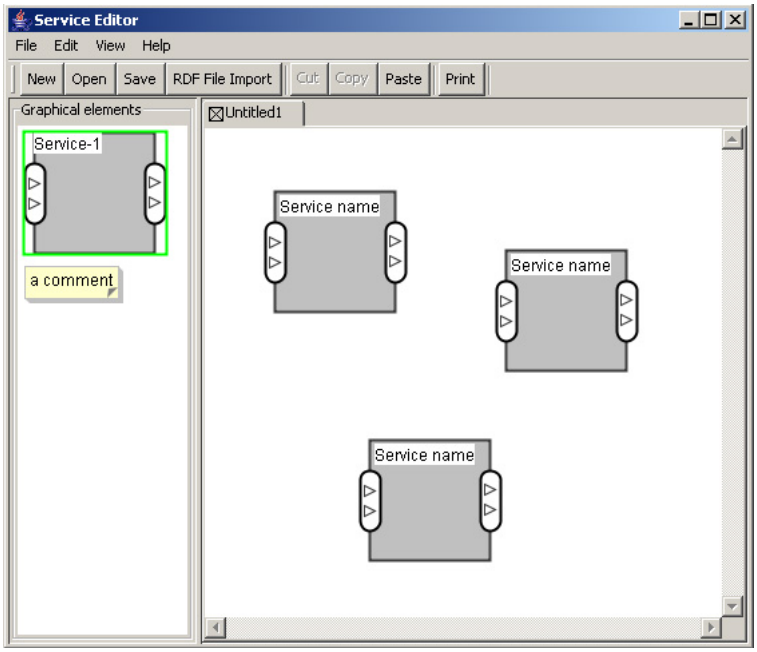

Figure 6.3: Service modeling tool: modeling service elements

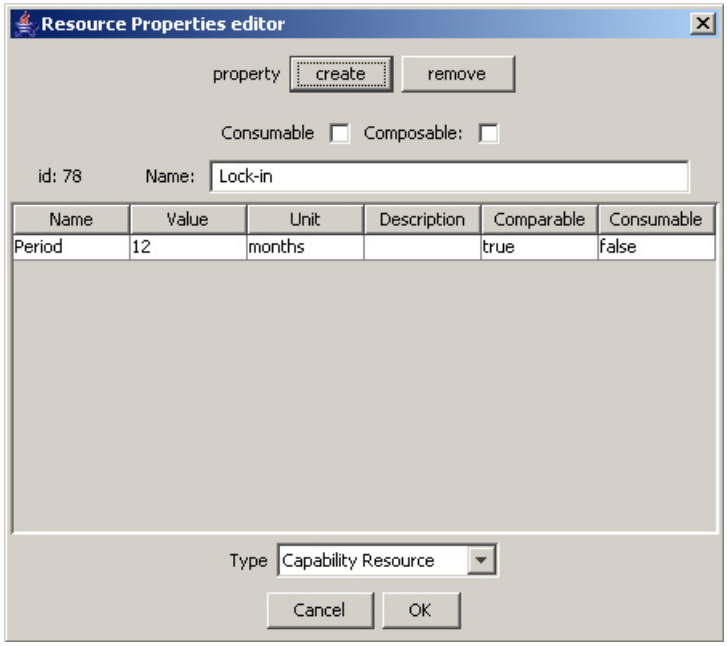

Figure 6.4: Service modeling tool: modeling a resource

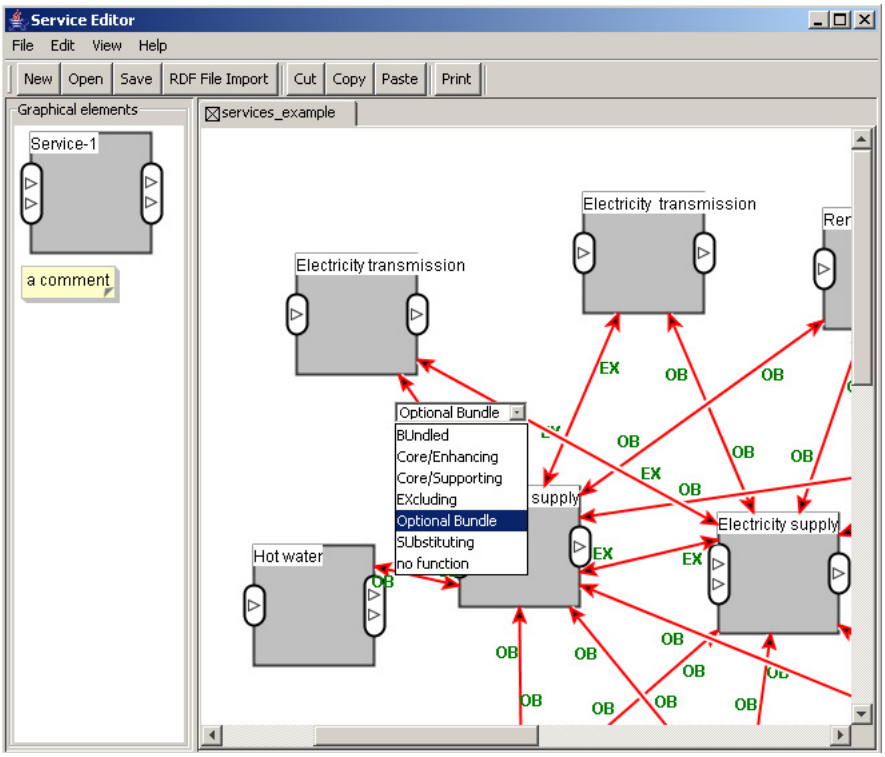

Figure 6.5: Service modeling tool: modeling service dependencies

set a requirement per service property (MINIMUM, MAXIMUM, EQUAL) (see Figure 6.6), or require the availability of a resource without setting requirements on its service properties.

Next, users can initiate the service configuration algorithm using the menu option 'File/Configure (with xsvg)'. This will trigger the configuration process described earlier. The whole communication between the service modeling tool, the configuration tool and the transformation module is invisible to the user. If the configuration algorithm terminates successfully, the generated service bundles will be visualized (see Figure 6.7). Otherwise, an error message appears.

In Chapter 7 we present a method for performing a business analysis using the *serviguration* ontology and the e^3*-value* ontology for business modeling. The method is supported by our service tool and by the e^3*-value* software tool which can be downloaded from `www.cs.vu.nl/~gordijn`. To facilitate software support for business analyses, we also implemented an interface between the service modeling tool and the e^3*-value* tool. This interface, embedded in the e^3*-value* tool, makes it possible to derive services from a value model (designed using the e^3*-value* tool), so that the service tool can be used for service bundling/configuration. Resulting service

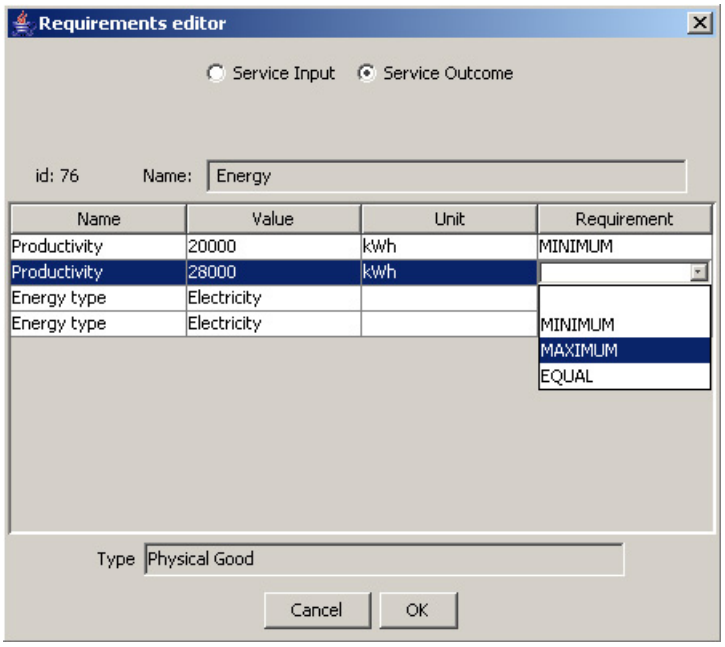

Figure 6.6: Service modeling tool: modeling bundling requirements

bundles can then be imported by the e^3-*value* tool to calculate financial feasibility. Information exchange between the two applications takes place using RDF files of both ontologies.

6.1.3 OBELIX Tool Support: Contribution to our Research

The service tool has been developed *during* our research, as part of the research effort. As such, it was not intended to represent a final product, but to facilitate the process that lead to the final product: the *serviguration* service ontology. This implies that the service tool uses an older version of the service ontology than presented in this thesis. While the major part of the ontology has remained stable, studies that we performed using the service tool showed that several changes and additions were required. These changes were made to the ontology, but the tool was not updated due to a lack of financial resources to do so, once the OBELIX project was terminated.

Three goals have been achieved with the service tool. First, the underlying ontology has been tested to be computationally adequate. Development of the service ontology and of the service tool was one process, rather than two separate ones. While designing and implementing the service tool we were faced with algorithms that required us to make minor changes in the ontology so that the necessary computing can be

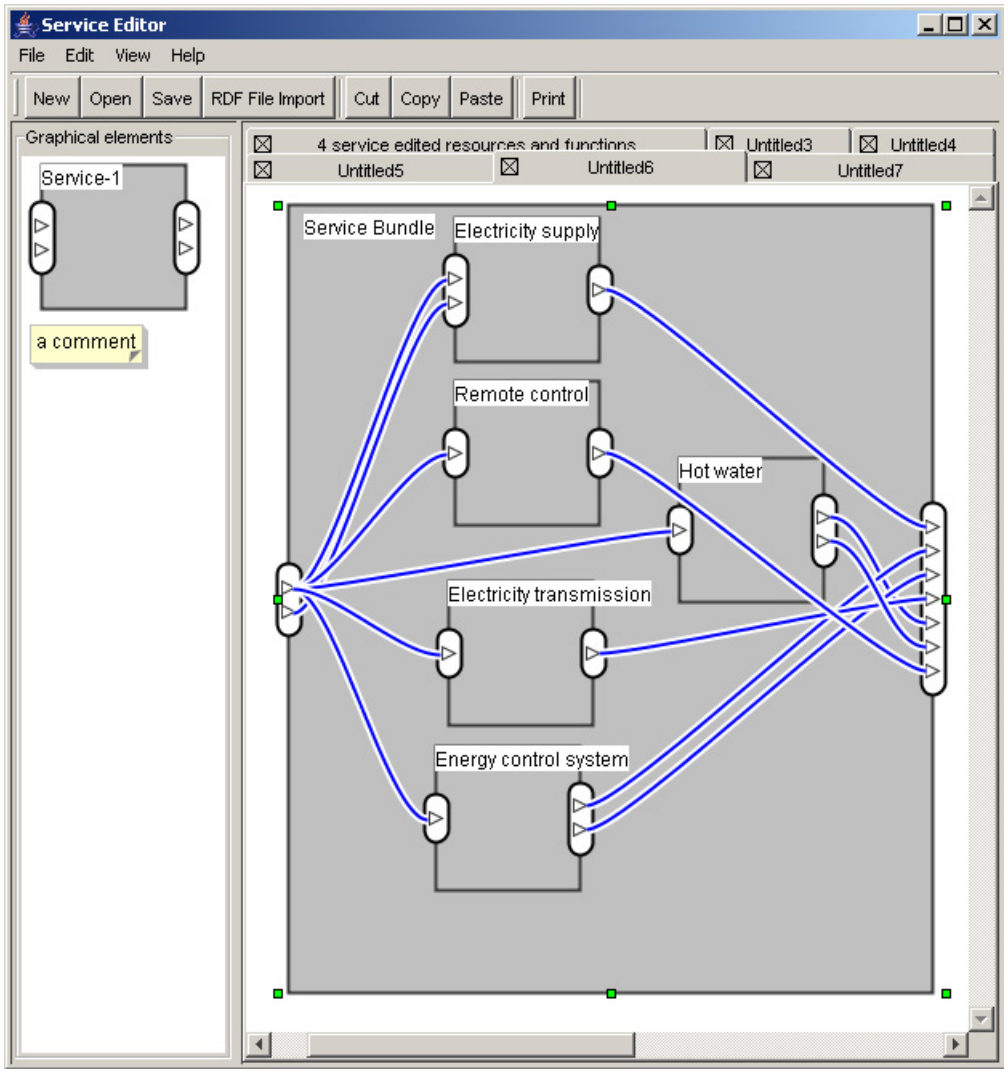

Figure 6.7: Service modeling tool: a visualized service bundle

performed. The final service tool demonstrated that required computations can be performed based on the underlying service ontology.

Test cases that we carried out to validate the output of the service tool (with and without the configuration module) showed that the tool generates correct results (as verified with domain experts). Hence also the second goal, validating the theoretical adequacy of the underlying ontology, has been achieved.

Third, modeling services requires domain knowledge, typically possessed by business people and other domain experts who are mostly not accustomed to formal conceptual modeling practices. Yet, they are required to model their knowledge in order to use model-based approaches as the one presented in this thesis. Before we had the service tool, we used ontology editors for modeling real-world cases. Even with modest datasets, as soon as several instances of the same concepts exist, manually administrating references to these instances becomes a fatiguing and error-prone task, because there is no automated mechanism to uniquely identify instances of concepts in a way that (1) the unique ID suggests how this instance is different from other instances of the same concept, and (2) a meaningful reference is given to other instances of other concepts to which this instance and concept is related in the ontology. The service tool made it possible for us and for domain experts to model larger studies.

6.1.4 OBELIX Tool Support: Drawbacks

Although the service tool served us well in our research, it has important drawbacks. First, it is based on an earlier version of the service ontology. The ontology has been updated based on conclusions from several studies, but the tool was not updated. Second, only the basic functionality required to achieve the earlier mentioned goals was implemented in the service tool. To make the tool fully operational, extra functionalities need to be added, mainly consistency checks (to verify that the domain expert does not model, for example, contradicting service dependencies) and an improved module for visualizing solution service bundles, that are imported from an RDF file. Third, performance has not been considered as a main issue in developing the service tool, because it was not required to achieve the goals of the tool, and because our ontology has not been influenced by considerations of computational complexity, as explained in Section 3.2.3. An operational tool will have to take performance issues into consideration, especially if the tool is to be used by customers via a website. Fourth, only the service offering perspective of the service ontology has been implemented in the tool. Tool support for the service value perspective will be discussed in the next section. We are currently involved in an extensive project in the health sector in The Netherlands, co-funded by the Dutch Ministry of Economic Affairs. As part of this project, called FrUX (Freeband User eXperience), a software tool will be developed for service bundling. This tool will tackle at least some of the drawbacks

mentioned here.

6.2 Tool Support for the Customer Perspective

6.2.1 Envisioned Tool Support: the FrUX Project

The OBELIX service tool requires that bundling (configuration) requirements are described in supplier terminology, i.e., in resources that services deliver. However, customers describe their demands in their own terminology, as captured by the service value (customer) perspective of our service ontology. To complete the automation of the *serviguration* process, software is required to model also the customer perspective, as well as the transformation process between the two perspectives, using production rules. Such tool support is being developed at the time of writing this thesis, as part of the Dutch FrUX project (Freeband User eXperience, `http://frux.freeband.nl`) co-funded by the Dutch Ministry of Economic Affairs.

FrUX' objectives include developing a Web-based information system, called DEM-DISC (Dröes et al. 2005). It will offer dementia patients and their (in)formal carers service bundles that suit their situation and are designed based on their specific needs. Although the project for which the tool is being developed specifically deals with health care, the underlying service ontology is generic, and hence the tool's engine could just as well be used for other services than health care and welfare services.

Unlike the OBELIX tool, which was intended for domain experts and modeling experts only, the FrUX tool will have a larger scope, and serve a variety of user groups. Besides domain experts, its users include also commercial service providers (who would have to model their service offerings), professionals (e.g., doctors), and most importantly: people with dementia and their informal carers. One of the issues in developing DEM-DISC is human computer interaction: how shall the information system interact with its diverse user groups, how shall knowledge be extracted from users, and how shall knowledge be presented to users. This research area is beyond the scope of this thesis, and is therefore not discussed further here.

6.2.2 Customer and Supplier Perspectives Integrated to Offer Service Bundles

A main disadvantage of the OBELIX tool is that it includes only a supplier perspective, making the tool not suitable for use by end users. The envisioned DEM-DISC will address this shortcoming, and include the customer and supplier perspective. As shown in Figure 6.8, it will provide the functionalities listed below.

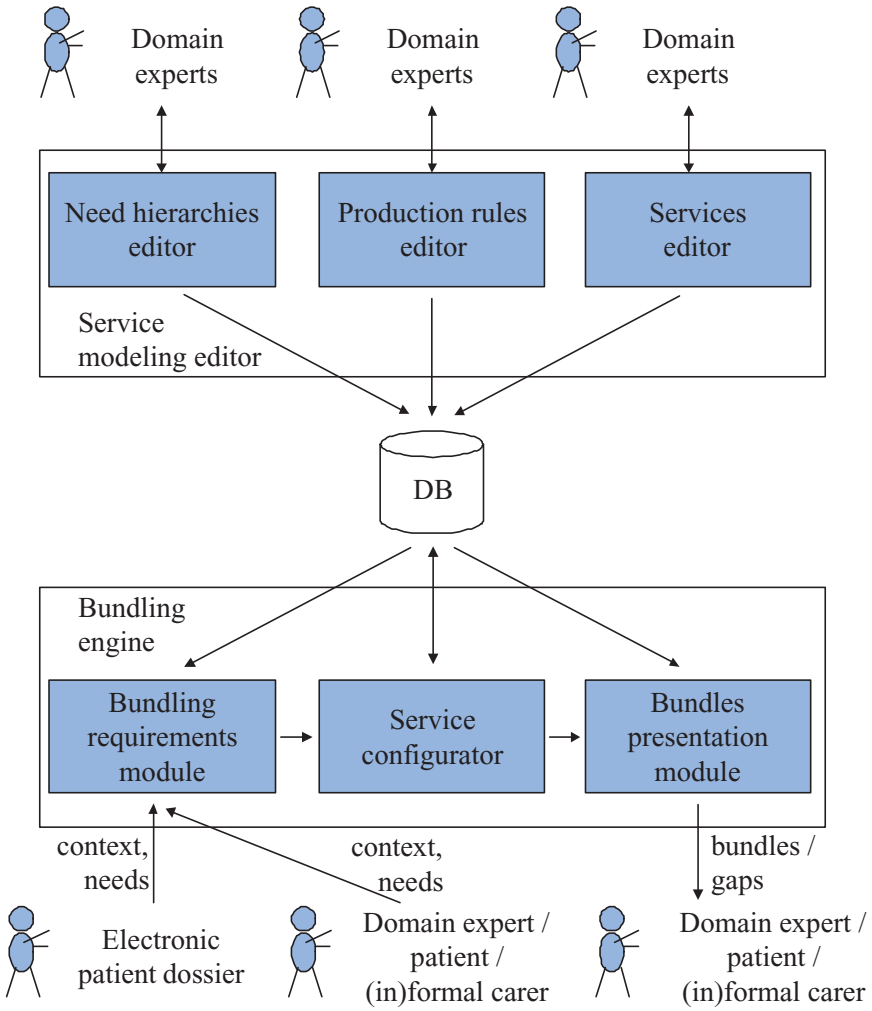

Figure 6.8: The envisioned DEM-DISC application. Arrows describe flow of information.

The most important functionality serves end users: patients, (in)formal carers and (semi) professionals:

- End users can describe their demands (based on earlier modeled need hierarchies) through an interactive human computer interface, and trigger the *serviguration* process (that uses knowledge modeled earlier by domain experts), resulting in zero or more service bundles that match the customer demands. These will then be presented to customers.

Other functionalities are required in order to provide DEM-DISC with domain knowledge that its reasoning engine uses, to achieve the above end users functionality:

- Domain experts can model need hierarchies for various types of customers (for commercial services, this knowledge may be available within marketing departments).

- Service providers or service brokers (we consider both to be sub-groups of 'domain experts') can model their available service offerings and business rules for combining services into bundles (*service dependencies*).

- Domain experts can define production rules for transforming customer terminology (demands) to supplier terminology (resources, outcomes of services).

- Domain experts can model how customer context information influences the *serviguration* process by defining relations between context elements and service elements and production rules.

- Domain experts can analyze possible service bundles and identify gaps in service offerings by describing possible customer demands and triggering the *serviguration* process. Solutions can be analyzed by domain experts. Where there are no solutions for a given set of customer demands, there is a gap in the service offering; domain experts can recommend to develop new service offerings to fill such gaps.

- Whenever domain experts model knowledge, a checker module verifies that the model adheres to constraints set by the service ontology. Here we do not refer to constraints on service bundling (configuration), but to inherent constraints on the usage of concepts and relations of the service ontology. For example it is forbidden to model two service dependencies between services A and B, such that core/supporting(A, B) and core/supporting(B, A). A list of constraints is given in Appendix A.

6.2.3 FrUX Tool Support: Discussion

DEM-DISC's functionality will enable domain experts model both perspectives on services: the customer perspective and the supplier perspective, such that customer-tailored service bundles can be generated by a software-based *serviguration* process.

DEM-DISC has a dual goal within FrUX. On the one hand, its prototypes are part of a research effort, helping us to validate our service ontology. On the other hand, the final information system is intended to actually offer service bundles to customers. As the software is not yet available at the time of writing this thesis, we imitated the *serviguration* process by means of pen and paper, using production rules and service dependencies to simulate the whole *serviguration* process starting with customer demands and ending with service bundles (we could not use the OBELIX tool, as it does not handle customer needs and the transformation between customer and supplier perspective). We discuss this validation of the ontology's theoretical adequacy in a study description in Chapter 8. As for DEM-DISC's role for communication with domain experts, we cannot yet assess DEM-DISC's contribution, as it is not yet available. However, the modeling effort performed by us with domain experts, carried out on paper, showed to be an administratively very demanding task (whenever a change is made somewhere in the model, its implications have to be found manually and corrected throughout the model), and therefore also error prone. Automated support will make the modeling effort more effective and efficient.

DEM-DISC's functionalities will help solve several main shortcomings of the earlier discussed OBELIX tool. First, DEM-DISC will use the most updated version of the service ontology, so all lessons learned from studies will be implemented. Second, it will include the full functionality, as described in the previous section. Third, and most important, it will integrate a customer perspective with a supplier perspective, while the OBELIX tool was limited to the supplier perspective only. Performance will also be monitored and considered during the development process, because an online application typically requires fast responses.

In spite of its major advantages, also the envisioned DEM-DISC application will not be complete, as it serves a specific goal, so its scope is limited to that goal. DEM-DISC's main goal is offering customer need driven service bundles to end users over the Internet. This is one of the usages of our service ontology, presented in this thesis (see Section 2.3.3). Another usage of the service ontology is performing a business analysis, as described in Section 2.3.4 and exemplified in Chapter 7. DEM-DISC will facilitate a business analysis only partially: it will enable domain experts to identify gaps in service offerings, by identifying demands for which no service offerings exists. As its goal is not to perform a financial analysis of business models, it is not planned to include an interface to other tools for investigating financial feasibility of service bundles. The latter is possible in the OBELIX tool, where an interface exists between the service modeling and configuration tool and the e^3*-value* business

modeling tool (for more details on this interface see Chapter 7).

6.3 Concluding Remarks

In this chapter we discussed two software tools for modeling service domains based on our service ontology. We implemented a first tool within the OBELIX project, and a second tool is being developed within the FrUX project at the time of writing this thesis. These ontology-based tools differ substantially from ontology editing tools as *Protégé* or *OntoEdit*.

First, while ontology editors are suitable for modeling any ontology, our tools are limited to modeling instances of our *serviguration* ontology only, because they are not intended to be generic ontology editing tools.

Second, we use our ontology-based software tools to model large-scale scenarios. Such large scenarios are hard to model using ontology editors, because instances of concepts need to be managed manually and based on IDs that are often not user friendly. Our software tools hide these technical details from users.

Third, and most importantly, the tools are designed for different target groups. *Protégé* and *OntoEdit* assume that users have some level of familiarity with structured modeling techniques. Our software tools do not make this assumption, because end-users are typically professionals (e.g., care professionals, consultants), business people (e.g., employees of service providers) and end-user customers (every person who consumes services). Consequently, ontology editors have a user interface which is suitable for information analysts and knowledge modeling experts, but not for our target group.

Our experience in multiple studies showed how important user friendly software tools are in studies involving stakeholders from a business or professional organization. Software tools play a dual role in these studies. First, they hide technical details from domain experts, thereby helping extract implicit domain knowledge from domain experts, and making this knowledge explicit. Second, they serve as a means to communicate ideas to stakeholders from a business or professional organization by visualizing conceptual models.

An anecdote may express the importance of such visual tools. During a review of the OBELIX project, the author of this thesis gave a presentation about the service ontology. The presentation started with an explanation of the various concepts of the ontology, together with screenshots that show how we use ontology editors to model services, and finally a demonstration of the service tool. As a reaction on the service tool demonstration, one of the reviewers, a gentleman from a Nordic business school, said: “Now I understand you; finally you talk business to me”. He did not understand the technical discussion that preceded the demonstration, because

it required a different way of thinking and reasoning than his. The service tool, where the ontology is hidden under a visual layer, proved to be a suitable way to discuss the underlying theory with him.

Part III

Ontology Application

Chapter 7

Energy Services

***Note**: This chapter demonstrates the use of the service ontology through a study in the energy sector, and includes a four steps method for performing a business analysis for finding financially interesting cooperations between service suppliers who can offer their independent services as a bundled offering. This chapter was published as papers in the proceedings of the 17th Bled eCommerce Conference (Baida, Gordijn, Morch, Sæle & Akkermans 2004) and in the proceedings of the 8th IASTED International Conference on Power and Energy Systems (PES 2005) (Morch, Sæle, Baida & Foss 2005).*

In Section 2.3 we presented two usages of our service ontology: as a business analysis tool and a conceptual model for software that realizes a web-portal through which customer-tailored cross-organizational service bundles can be offered to customers. The current chapter exemplifies the first of these two usages.

Model-based approaches to developing cross-organizational e-business initiatives help involved enterprises understand the initiatives by creating a shared understanding as a basis for profitability assessment. Still, when developing business models where multiple potential enterprises may participate in offering a service bundle, complexity increases. Our study partner, the electric utility TrønderEnergi AS from Trondheim, Norway, understood that structured, computer-based techniques can help reduce this complexity, and joined forces with us in employing computer-based techniques to perform a business analysis and develop possible future business models. The question at hand was which financially feasible service bundles to offer to customers, such that electricity supply is bundled with additional services? The choice to market specific service bundles is in fact a choice for specific business models.

Software-aided reasoning processes can provide business developers support in the selection of services to include in service bundles, implying also a selection of partners to work with. To put it differently, given a set of potential services to include

in a new business model and dependencies between these services, we need tools to design (configure) one or more service bundles, and to provide information for assessing the pros and cons of service bundles. Then the business analysis can continue by calculating profitability of these service bundles. The configuration process takes into consideration inherent dependencies between available services and possibly other requirements related to service properties as price, quality and more.

To this end, the service ontology presented in this thesis can be used with an ontology for designing business models, for instance the e^3-*value* ontology. In the current chapter we present an extensive study in which we combined the two ontologies to perform such a study for TrønderEnergi AS. We concentrate on the part of the study that deals with the service offering perspective of our service ontology. From a business perspective, the goal of TrønderEnergi AS was to enhance understanding of possible new business models for bundling electricity with other services. From an ontology development perspective, the study was used for ontology validation.

7.1 E-Services in the Energy Sector

7.1.1 Introduction to the Energy Sector

Since the deregulation of the electricity market in Norway in 1991, production and trade of electric energy have been liberalized, while the transmission and distribution are maintained as regulated monopolies. Nowadays, after evolving for 15 years of deregulation, the Norwegian power market becomes mature. The electricity generation and supply sectors are characterized by a fierce competition, due to which the difference in electricity retail prices per kWh between different suppliers is diminishing. Also in other European countries power is shifting from suppliers to customers, and more and more end-user customers in Europe are able to choose a preferred electricity supplier.

Commercially, one of the disadvantages of the electricity product is that for power supply companies it is hard to distinguish themselves, due to the anonymous nature of this product: electricity from different suppliers is delivered according to the same standard, with the same physical characteristics, and is consumed through the same electricity socket in a customer's home. Therefore, companies face difficulties in competing with each other. Consequently, many suppliers are seeking for ways to improve marketing via differentiation of their product, to increase their market share. One way to differentiate is to offer additional services such as Internet access, (software) application service provisioning and home comfort management. Product differentiation can also be achieved by introducing substitutes as "green energy". Another way to improve marketing is to create more complex and elaborated electricity retail contracts, which are more beneficial to customers because they fit better to their

needs. At the same time, choosing the best electricity contract becomes a demanding task for electricity consumers.

Many of the additional services can be ordered and provisioned via the Internet. Moreover suppliers can use existing infrastructure and/or available business processes to deploy such extra services, so bundling these services with the traditional electricity product can be done with relatively modest effort. Experience however shows that the bundling of services without sound logical fundaments of the bundles design process and disregarding customers' demands may cause severe financial losses, as can be seen by the example of KanKan (Flæte & Ottesen 2001). KanKan was launched on January 23^{rd} 2001 as a new market offer of one of the biggest Distribution System Operators in Norway. It was marketed as an integrated bundle of services, including electricity supply and transmission, Smart Home features, home insurance, telephone and an ISP service. Despite the expectations and costly market campaigns, very few households showed interest in the new service offering. After several attempts to revise the concept, it was removed from the market (Flæte & Ottesen 2001, Martinussen 2002). Several reasons for the failure were identified later; misunderstanding of customer needs and meeting them in product offers was the most visible one. Furthermore, the KanKan example highlights the necessity for evaluation methods for the feasibility of offering service bundles.

7.1.2 TrønderEnergi AS

Following the deregulation TrønderEnergi AS as many other electric utilities in Norway was re-organized into a holding company with various subsidiaries. The company has become a strong player on the Norwegian market, selling traditional electricity-related core products as well as new products, for example providing broadband Internet access or delivery of hot water (for room heating) to customers. In 2004 TrønderEnergi AS had a revenue of 656,927,000 NOK (TrønderEnergi AS 2004), equaling about 83,000,000 Euro. The company generated 1723 GWh electricity (the average annual generation is 1831 GWh) of the 110.1 TWh electricity that was produced in Norway that year (Statnett SF 2005). The company wants to use the new corporate structure, which is shown in Figure 7.1, in order to improve its position on the market. TrønderEnergi AS however, is aware of potential financial risks related to implementation of a wrong bundling strategy. Although several of the subsidiaries within the holding company are economically independent (they are responsible for their own profit and loss), the mother company's interest is to utilize the various service offerings in order to offer service bundles where the electricity product (sold by one subsidiary) is differentiated by additional services (sold by other subsidiaries).

The example of KanKan along with several similar cases related to bundling makes the mother company very cautious and sceptic when it comes to implementation

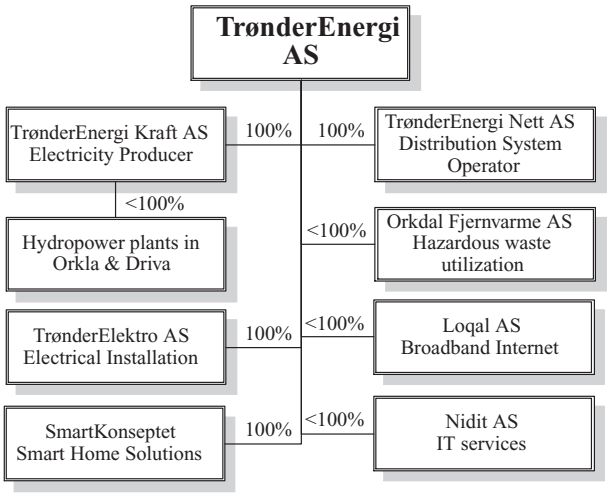

Figure 7.1: TrønderEnergi AS corporate structure (percentages show the share of the mother company)

of bundled offerings. In our study we performed a thorough analysis of the services which can be offered by the holding company, including their pricing, possible composition of bundles, probable limitations and potential benefits. We also evaluated the financial feasibility of different compositions of bundles. For example, we assessed whether offering a heat pump along with an electricity retail contract to a customer is going to be profitable for the company or not. The study presented in this chapter utilizes and exemplifies our service ontology, as well as a value ontology for business models design (Gordijn & Akkermans 2001, Gordijn & Akkermans 2003*b*). A comprehensive project report of the study at hand can be downloaded from `www.cs.vu.nl/~obelix/D7.2.pdf`.

7.2 A Four Steps Method for Business Analysis

The e^3-*value* method (Gordijn & Akkermans 2001, Gordijn & Akkermans 2003*b*) – based on the e^3-*value* ontology – is an established multi-actor approach for developing e-business models, taking into consideration the importance of economic value for all actors involved, and the intertwining of business and technology. Similar to the service ontology, it is based on the understanding that business activities involve an exchange of economic values between the involved actors.

When applied to the service sector we found that an e^3-*value* business model does not provide a logical framework for reasoning about how to bundle services. Such a business model cannot describe in detail the variety and complicated nature of po-

tential service bundles. Nor does it handle inherent dependencies between multiple services, such as 'service X may not be offered without service Y'. This information is necessary in order to configure feasible service bundles and to point out differences between and redundancies among possible service bundles. Thus, we need extra information on services, to facilitate a complete business model analysis of service offerings. Consequently, we suggest using the *e*^{3}*-value* ontology with our *serviguration* ontology that provides a conceptualization of special service characteristics, not present in a value ontology. Our service ontology, presented in this thesis, provides a conceptualization of services, seen as components that require some inputs and provide some outcomes. Dependencies between services are also formalized, providing a mechanism for reasoning about which services must or may be part of a service bundle, and why. Using both ontologies together enables us to evaluate complex service offering scenarios. Our method includes the following steps:

1. Create an initial business model, using the *e*^{3}*-value* ontology. Elementary services can be identified in this model.
2. Model these services using the *serviguration* ontology, and define service bundles by applying a configuration algorithm.
3. Reason about the identified service bundles, using knowledge modeled in the *serviguration* ontology, and choose the preferred service bundles.
4. Use the *e*^{3}*-value* ontology to assess profitability of the chosen service bundles.

In the first step and in the last one we use a value ontology. The added value of using and applying the service ontology in steps two and three is that step four becomes manageable: we only assess profitability of service bundles that have been identified as interesting in steps 2 and 3.

We chose the *e*^{3}*-value* ontology as a starting point, rather than another value ontology, mainly because it is designed for modeling multi-enterprise scenarios. Other value ontologies as the AIAI enterprise ontology (Uschold et al. 1998), the Toronto Virtual Enterprise Ontology (TOVE) (Fox & Gruninger 1998) and the Business Model Ontology (BMO) (Osterwalder & Pigneur 2002, Osterwalder 2004) focus on a single enterprise. The Resource Event Agent (REA) ontology (McCarthy 1982, Geerts & McCarthy 1999) is not an approach for business development, from a methodological point of view, and is therefore not a suitable candidate for the current study.

7.3 Step 1: A Value Model for Energy Services

A first step in creating a multi-enterprise business model is to understand the elementary services that are possible. In many cases, these services cannot be easily enu-

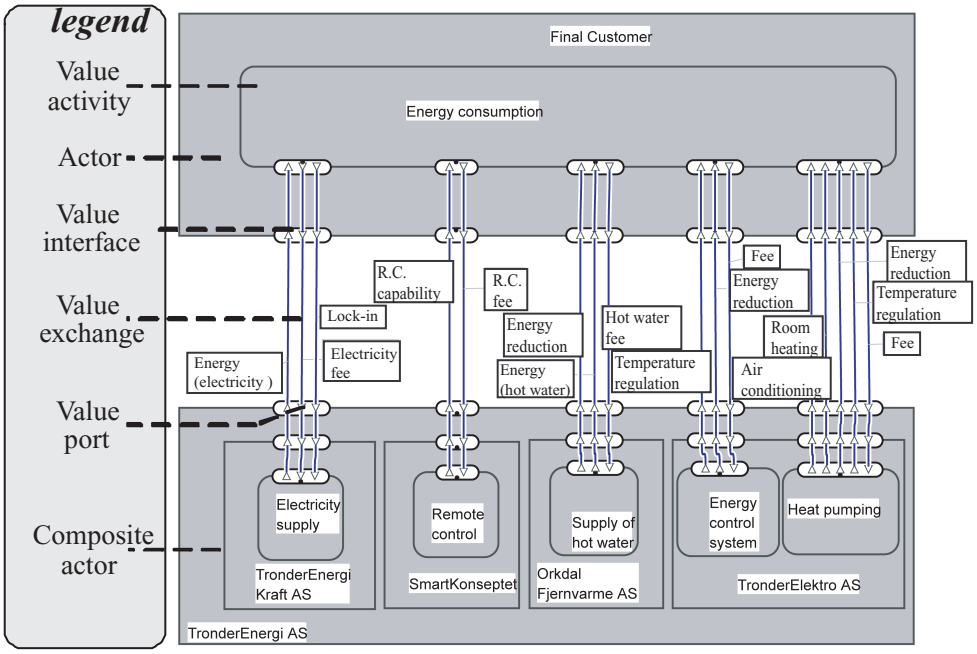

Figure 7.2: Initial value model for energy services

merated because stakeholders do not have a clear view on such services. To this end, we constructed an e^3-*value* model (see Figure 7.2) that shows the services enterprises are offering to customers, as well as what they request in return. The construction of such a model involves eliciting services that exist in reality or that stakeholders want to develop. The e^3-*value* method has been discussed extensively in Gordijn & Akkermans (2001) and Gordijn & Akkermans (2003*b*) , so we only present the model itself. Due to model complexity, we only present a fraction of the model here. Figure 7.2 shows a number of actors: an end-user customer and a number of enterprises offering a range of services (e.g., TrønderEnergi Kraft AS and Smartkonseptet).

Actors exchange *value objects*, objects of economic value such as physical objects or fees. We model only things of economic value, and not information required for business processes. This ensures that stakeholders concentrate on understanding the values offered and requested, and nothing else. Examples of value objects in this case are electricity, the capability of remote control of devices such as heaters or coolers, and the capabilities for energy consumption control and temperature regulation. Fees are value objects too.

Value objects are offered and requested via value ports, depicted by triangles. A triangle shows whether a particular actor requests or delivers an object of value to or from its environment. Ports are grouped into value interfaces, depicted by small rounded

boxes surrounding two or more value ports. Such a value interface fulfills two modeling purposes. First, a value interface models economic reciprocity. For instance, it says that electricity is delivered only if a fee is paid in return, and vice versa. Second, a value interface may represent bundling, saying that two or more value objects are offered (or requested) only in combination. Figure 7.2 does deliberately not represent such a bundling case. In this chapter we discuss how to find such bundles for known elementary services.

Value ports are connected by value exchanges, represented by lines. Exchanges represent that actors are willing to exchange objects of value with each other. Finally, rounded rectangles represent value activities. These are activities that are supposed to be profitable for at least one actor. The main rationale for such activities is to distinguish actors (enterprises) from what they are doing to make profit (value activities). The use of the e^3-*value* method in networked enterprise analyses has shown that the discussion on 'who does what' reflects an important business design decision.

The value model in Figure 7.2 shows actors, activities they perform, objects of value they offer and what they request in return, but not which meaningful bundles of value objects can be constructed. In a complex value model with many actors and value objects, finding these bundles is a far from trivial task. Moreover, the e^3-*value* ontology is not of help here, since it does not model considerations to bundle objects (or to exclude certain bundles); it only models the bundles themselves. To this end, we propose the *serviguration* ontology that connects well to the e^3-*value* ontology, with the aim to assist in finding such bundles specifically for services.

7.4 Step 2: A Service Model for Energy Services

The service ontology presented in earlier chapters formally describes a shared view on what services are with the aim to compose (or: configure) complex services out of more elementary services supplied by different enterprises.

Service elements are the building blocks of a service bundle. They represent what a supplier offers to its customers, in supplier terminology. It is what the business literature defines as service, an economic activity (performance) of mostly intangible nature (see Section 2.1.2). Elementary services result from our initial value model, depicted in Figure 7.2. Value activities in the e^3-*value* ontology correspond to service elements in the *serviguration* ontology. Additionally, value objects in the e^3-*value* ontology correspond to resources in the *serviguration* ontology.

We modeled 14 services that can be offered to customers in bundles that include energy supply: electricity supply, electricity transmission, hot water distribution (for room heating), broadband Internet access, IT-services, sales and installation of electrical appliances (heat pump and energy control system, to reduce energy consump-

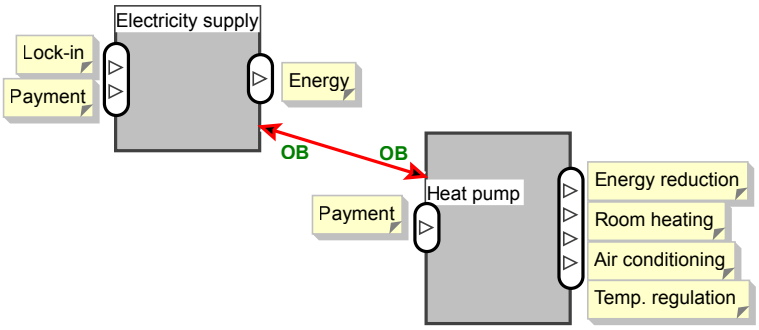

Figure 7.3: Example service elements: electricity supply and heat pumping. Service inputs are shown on the left side of a service element, and service outcomes are shown on the right side thereof.

tion and to regulate temperature), temperature remote control and more. Some services were modeled multiple times, as they can be provisioned in different forms.

Figure 7.3 is a partial visualization of two service elements: electricity supply and heat pumping. The symbols 'OB' mark service dependencies between the involved service elements: Optional Bundle. This service dependency can be interpreted as 'there is business logic in bundling these services, but they may also be provisioned independently'. Service inputs are listed on the left hand side of a service; service outcomes are listed on the right hand side thereof. A more comprehensive list of services, described by their service inputs and by their service outcomes, is presented in Table 7.1.

Figure 7.4 is a visualization of seven of the fourteen services we modeled in this study. Three service elements (electricity supply, electricity transmission and energy control system) were modeled twice because they are available for consumption in different forms, resulting in a set of ten services. This set serves us for the current discussion; it is the same set of services as presented in Table 7.1.

The 'base' e^3-*value* model in Figure 7.2 did not consider dependencies between different service elements, giving no guidelines on how to combine services into a service bundle. Theoretically, we could design and assess a business model for any combination of one or more services, making the development of financial calculations for the model very time-consuming due to the multitude of possible solutions. With a set of n service elements, and assuming that a service bundle may include one or more service elements and (for simplicity) that no service is included more than once in a bundle, as many as $2^n - 1$ distinct service bundles are theoretically possible.[1] Using a set of ten services, this yields 1023 possible bundles. However,

[1] In terms of the configuration algorithm explained in Chapter 4, the formula $2^n - 1$ assumes that for every high-level configuration there exists one detail-level configuration; in reality the number of

Table 7.1: Energy services described by their service input resources and by their service outcome resources (resource type is mentioned in brackets)

Service name	*Service inputs*	*Service outcomes*
electricity supply	fee (monetary resource), lock-in (capability resource)	energy of type electricity (physical good resource)
electricity transmission	fee (monetary resource)	electricity transmission (state-change resource)
hot water supply	fee (monetary resource)	energy of type hot water (physical good resource), energy reduction (capability resource)
heat pump	fee (monetary resource)	energy reduction (capability resource), room heating (capability resource), air conditioning (capability resource), temperature regulation (capability resource)
energy control system	fee (monetary resource)	energy reduction (capability resource), temperature regulation (capability resource)
remote control	fee (monetary resource)	remote temperature control (capability resource)
broadband access	fee (monetary resource), lock-in (capability resource)	Internet connectivity (capability resource)

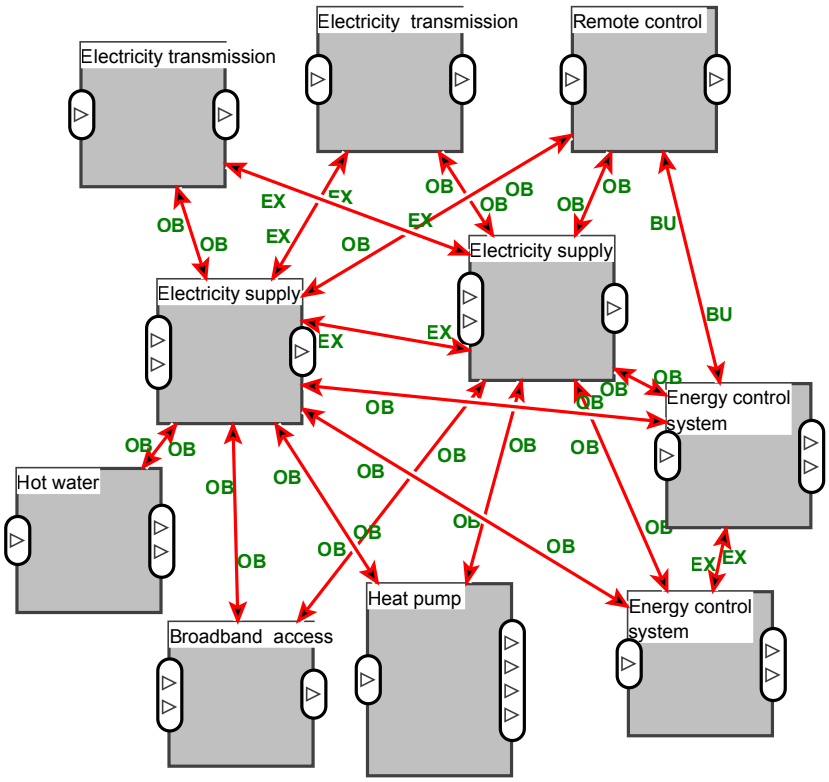

Figure 7.4: Service elements and their service dependencies

many of these bundles are not based on business logic, and therefore it is worthless to spend time analyzing their financial feasibility. The service ontology was applied to resolve this problem, narrowing the scope of our primary business model analysis:

1. Step 2 of our method: 1023 service bundles could theoretically be created using all given services. The service ontology identified those bundles that are driven by business logic, omitting all other theoretically possible bundles (step two in our method). Using our set of ten service elements, we generated bundles for five different scenarios, involving five different sets of bundling requirements (varying from very specific requirements to more general ones). Using the service ontology and the service configuration software tool, we reduced complexity to sets of only 2, 8 16, 17 and 28 service bundles for the five different scenarios. Our software tool was very helpful in this task, because generating these service bundles manually, with no tool support, is an error prone task.

detail-level solutions per high-level solution may be higher.

2. Step 3 of our method: Providing knowledge on services, to facilitate a reasoning about a choice between the bundles that were designed in step two.

Services of subsidiaries may be bundled due to various reasons, including an efficient use of common business processes, interdependencies between services and more. In the service ontology we represent this knowledge as service dependencies, business rules that form constraints on whether and how two or more services may be combined into a service bundle. For example, some services can be sold only with other services; some services exclude others (they may not be sold as a bundle); and some services can be added to others, to make the other service more attractive. We modeled business rules concerning energy services as service dependencies, to be used as constraints in the software-aided service configuration process. These are listed in Table 7.2. The most often used service dependency in this table is 'optional bundle', implying that there is business logic behind combining two services into a bundle but the separate services can also be consumed independently of each other. Note that at the end of a business analysis, if a choice is made to market two services only as a package rather than also as elementary services, an 'optional bundle' dependency will be changed to a 'bundled' dependency, reflecting the new business decision. By applying service dependencies between service elements, we generated a set of service bundles, omitting bundles that have no business logic (from a supplier's point of view). Examples of possible bundles are:

1. Electricity supply and heat pumping
2. Electricity supply and hot water
3. Electricity supply, energy control system and remote control

No service dependency exists between the services *heat pumping* and *hot water*, because there is no business logic behind a bundle that includes only these two services (a heat pump reduces electricity consumption, but when hot water replaces all the use of electricity for heating, there is no electricity consumption to reduce). Consequently the bundle of *heat pumping* and *hot water* is irrelevant, and was not generated. On the other hand, since a 'bundled' service dependency between *remote control* and *energy control system* requires that *remote control* is not sold without *energy control system*, all possible bundles with *remote control* but without *energy control system* are invalid, and were not generated. This knowledge does not exist in a value model.

7.5 Step 3: Service Ontology for Business Analysis

In step three of our method we reason about theoretically feasible service bundles, and make a choice about preferred bundles. Our reasoning is based on the assumption that a supplier wishes to offer service bundles that satisfy its customer needs and

Table 7.2: Service dependencies in the energy study. A service dependency is a function with two arguments; the first argument is the service in the row, and the second argument is the service in the column. The abbreviations OB, EX and BU stand for the dependencies 'optional bundle', 'excluding' and 'bundled'. Some services are modeled twice because they can be provisioned in different forms.

	electricity supply (1)	electricity supply (2)	electricity transmission (1)	electricity transmission (2)	hot water supply	heat pump	energy control system (1)	energy control system (2)	remote control	broadband access
electricity supply (1)		EX	OB	EX	OB	OB	OB	OB	OB	OB
electricity supply (2)	EX		EX	OB		OB	OB	OB	OB	OB
electricity transmission (1)	OB	EX								
electricity transmission (2)	EX	OB								
hot water supply	OB									
heat pump	OB	OB								
energy control system (1)	OB	OB						EX		
energy control system (2)	OB	OB					EX		BU	
remote control	OB	OB						BU		
broadband access	OB	OB								

demands. These are modeled in the service value perspective of our service ontology. Table 7.3 presents a hierarchy of customer needs, wants and demands for the study at hand. We modeled a hierarchy of customer needs, and defined relations between customer demands and service outcomes, descriptors of available services (see Figure 7.5). These relations have the form of 'IF demand-X THEN service outcome-Y' and implicitly 'IF service outcome-Y THEN service element-Z', reflecting a logical correlation: service element Z provides service outcome (resource) Y, which can satisfy demand X. Demands and resources can be described by quality criteria, such as productivity, availability and more.

Applying these relations results in sets of service bundles per customer demand. Based on knowledge that the service ontology provides, business developers then reason about these bundles. Some of them may appear to be redundant (because they compete with each other on satisfying the same customer demands). Others may be suitable only in certain circumstances (certain areas or customer types). A choice to offer certain bundles implies also a choice of business partners to work with.

To satisfy a customer demand for energy supply a bundle may theoretically include almost any combination of the following services: electricity supply, heat pumping and hot water (as well as other obligatory services that we do not discuss here). However, the service ontology provides extra tools to narrow the scope of our analysis:

- Hot water (replacing part of the electricity consumption, for a lower price) is available in a limited geographic area only; hence, different service offerings are possible in different areas. This is modeled in Figure 7.5: a context switch triggers different relations (production rules) between the demand for energy supply and the resources 'energy of type electricity' and 'energy of type hot water', based on the given zip code.

- Customers would prefer bundling electricity supply with hot water to bundling electricity supply with heat pumping due to a lower price.[2] Consequently, where the hot water service is available, offering electricity supply with heat pumping may be less attractive. The difference in price is modeled by the pricing model concept that is attached to the fee input of commercial services.

Let us now take a new customer demand into consideration: temperature regulation, for indoor comfort. The following service elements satisfy this demand for commercial customers: heat pumping, energy control system and remote control. Also here the service ontology provides extra information for our business analysis:

[2] A lower price is achieved only over time, because customers who wish to consume the hot water supply service for room heating are required to invest in hardware.

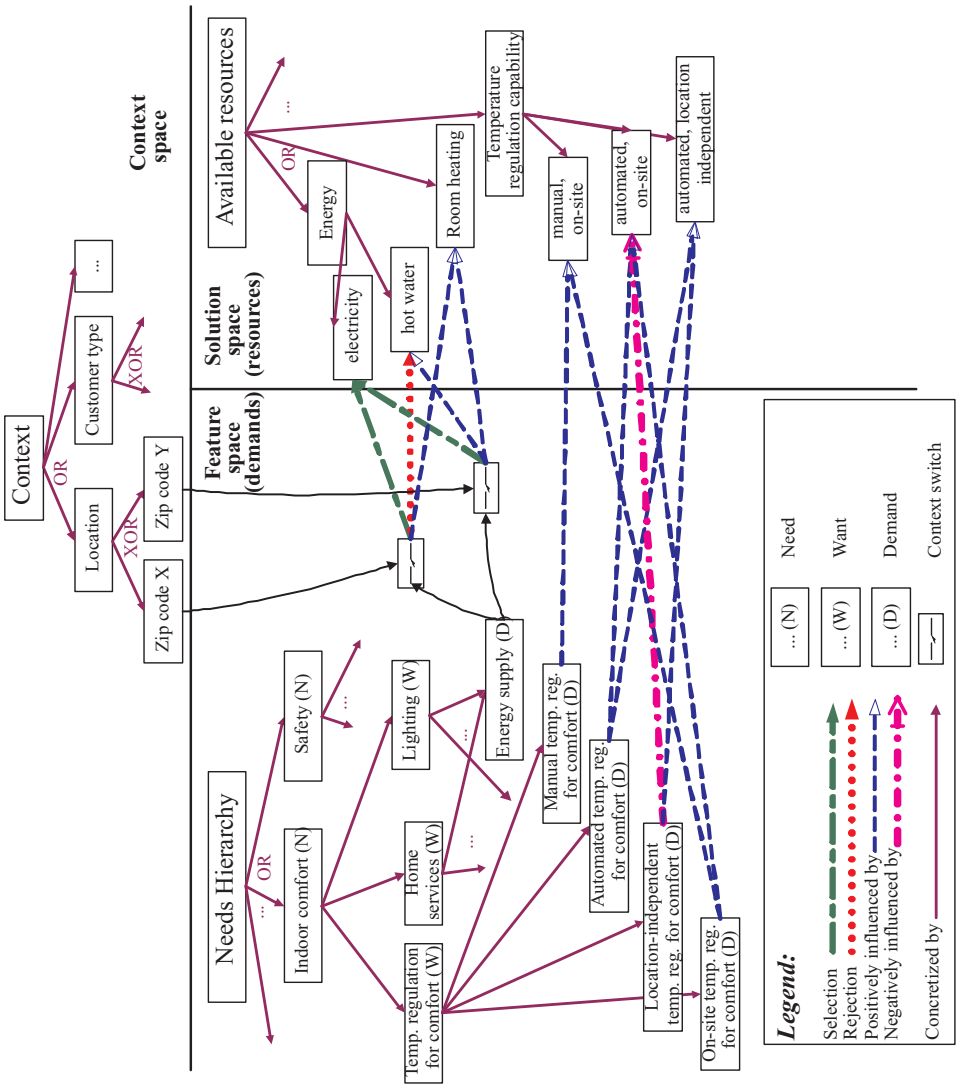

Figure 7.5: Partial FS-graph of the energy study: production rules model how resources can satisfy customer demands.

1. Manual and location-dependent (only on-site) temperature regulation requires the following service elements: electricity supply and heat pumping. If a customer already consumes these services for his energy supply, manual energy regulation is available with no extra costs (see the top left service bundle in Figure 7.6).

2. Automated and location-dependent (only on-site) temperature regulation requires the following service elements: electricity supply and energy control system (see the top right service bundle in Figure 7.6). Unlike the first bundle, this one does not provide the outcome "air conditioning".

3. Automated and location-independent (via a website) temperature regulation requires the following service elements: electricity supply, energy control system and remote control (it also requires an ISP service, but we omit this from the current discussion for brevity) (see the bottom service bundle in Figure 7.6).

Suppliers may then decide whether they want to offer all three bundles, or whether they want to profile themselves as online energy suppliers, and supply only the online temperature regulation version. If electrical appliances and remote control are offered by different companies, this implies also a choice of partners to work with. Although all three bundles satisfy the same customer needs and wants, as we have seen they are essentially different due to their properties. For our example let us assume that the choice was made to supply the third of these bundles.

7.6 Step 4: Value Ontology for Business Analysis

In the last step of our method we develop business models for the chosen bundles, and assess their profitability. Profitability assessment is not shown here (for a detailed explanation see Gordijn & Akkermans (2003*b*)), but only how found bundles can be fed back into an e^3-*value* model. All feasible bundles that were not chosen in step three are discarded, so their profitability need not be assessed. Chosen bundles can be shown in a revised e^3-*value* model (see Figure 7.7). In this case we restrict ourselves to bundle 3 as explained in the previous section. A customer demand as identified in the service ontology, e.g., automated, location independent temperature regulation, is represented by an e^3-*value* start stimulus. Such a stimulus shows the consumer demand, and connects to one or more value interfaces of the actor that has such a demand. The actor then exchanges objects of economic value to satisfy the demand via one of the connected value interfaces. In our case, the demand is connected to three interfaces via an AND-fork, saying that in order to satisfy the need, the actor must exchange objects via all three interfaces. Information elicited by using the service ontology was very useful when calculating profitability of the chosen bundles.

Table 7.3: Customer needs, wants and demands for the energy utility TrønderEnergi. The notations H/I refer to the customer type: Household or Industrial.

Customer Needs	*Customer Wants*	*Customer Demands*
Indoor comfort (H,I)	Lighting (H,I); Home services (cooking, washing) (H); Comfort temperature (H,I)	Energy supply (H,I); Hot tap water (H,I); Room heating (H,I); Air conditioning (H,I)
	Energy regulation for budget control (H,I)	Energy regulation for budget control (H,I), with different characteristics (manual / automated; on-site regulation / location-independent
	Temperature regulation for increased comfort (H,I)	Temperature regulation (H,I) with different characteristics (manual / automated, on-site regulation / location-independent)
Social contacts and Recreation (H); Business contacts (I)	Communication (H,I)	Telephone line (H,I); Mobile phone line (H,I); Internet (broadband) (H,I)
Safety (H,I)	Increased security (H,I); Reduced insurance premium (H)	Safety check of electrical installation (H); Internal control of electrical installation (I)
IT support for business (I)	IT-services (I)	ASP-services (I); Hardware (I); Software (I)

For example: in the initial e^3-*value* model it was difficult to define some value exchanges, because domain experts had to make assumptions, e.g., about the demand. The service ontology-based model allows us to verify the existing financial formulas and create the missing ones because it includes more details. Take for example the bundle that includes electricity supply and heat pumping: we can make a better assessment of electricity consumption (and thus the costs) during winter and summer for customers, because this information is modeled using the service ontology. We can derive very realistic figures, based on the composition of the bundle.

A found bundle in the previous section is represented in Figure 7.7 as a value interface for the composite actor TrønderEnergi AS that bundles ports exchanging a remote control service, electricity, and energy control. Additionally, the reciprocal value objects (fees, lock-in) are also shown in the value interface. Note that a value interface exactly models bundling: it is only possible to obtain the bundled services in combination, in return for the sacrifice stated. Other bundles can be modeled similarly. In the current study, step two generates at most dozens of feasible bundles, based on ten elementary services. In step three we choose only a subset thereof for profitability assessment. A recurring element in service bundles in step two is that the services 'electricity supply', 'remote control' and 'energy control system' are always bundled, while other services as 'heat pumping' and 'supply of hot water' are included in some of the bundles only. In accordance with these findings, business developers chose to investigate the financial feasibility of a business model where 'electricity supply', 'remote control' and 'energy control system' are marketed as a package, while 'heat pumping' and 'supply of hot water' may be sold separately. This choice is reflected in Figure 7.7. From a business development perspective, the choice to market several services as a bundle is an important decision. Investigating this option was a direct result of our service configuration approach.

7.7 Analysis and Conclusions

7.7.1 Business Perspective

Developing a multi-actor business model for e-service bundles involves various potential partners, each offering a number of services; only a subset of these services has to be selected for a business model. However, why choose for one service or another? Assessing profitability of all possible scenarios is often undesired, because it is a very time consuming task. In this chapter we presented how we use our service ontology together with a value ontology to tackle this business problem. When a broad spectrum of services is included in a business analysis, our ontology helps business developers design meaningful service bundles, and discard all other scenarios. As a result, the scope of financial feasibility studies remains manageable.

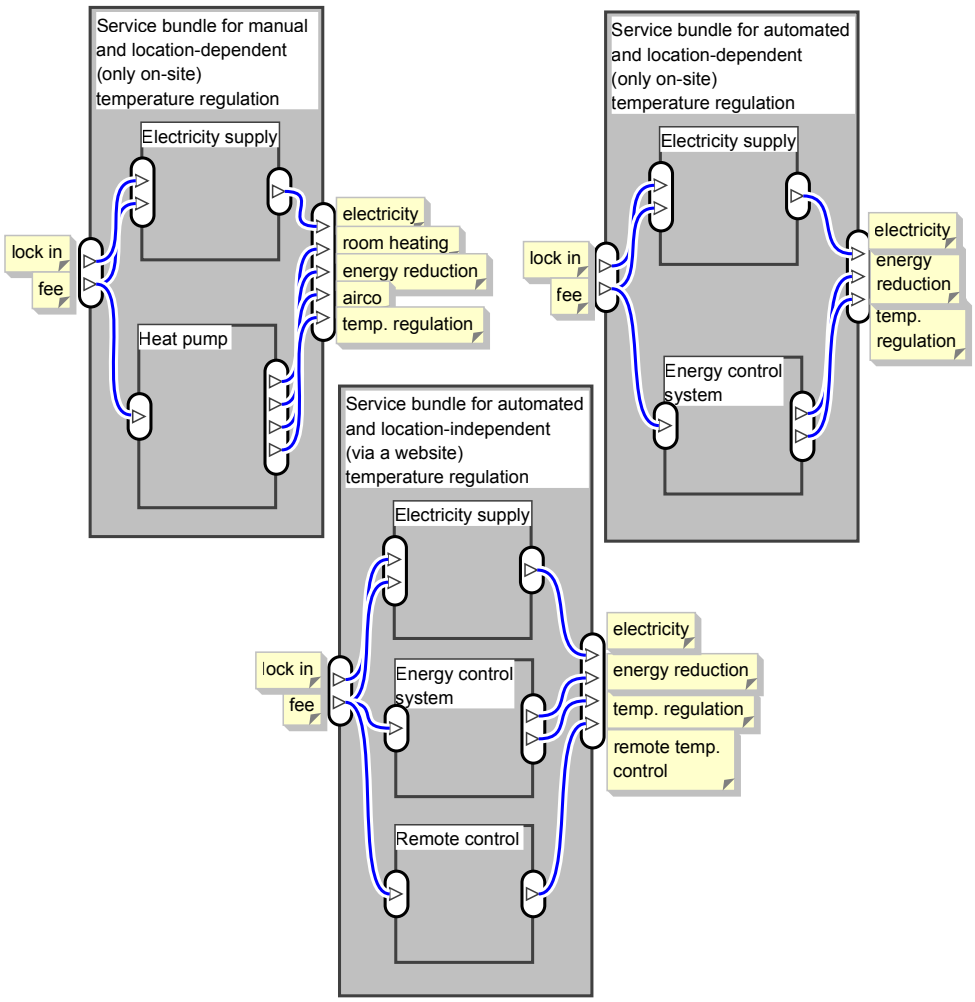

Figure 7.6: Step 3 of our method for performing business analysis yields three different service bundles for three similar customer demands

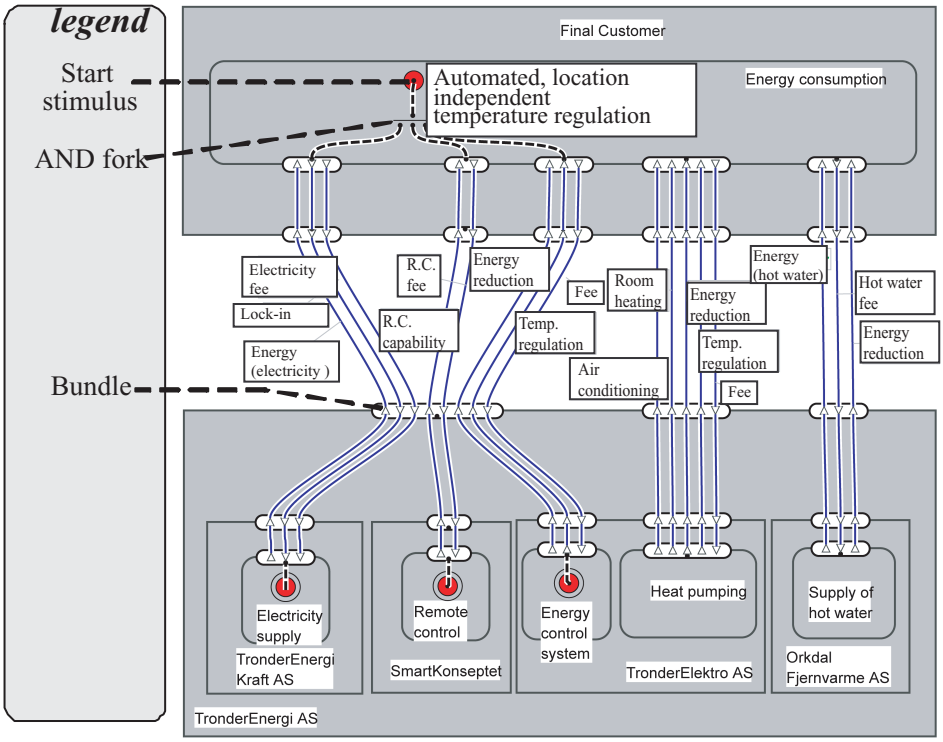

Figure 7.7: A revised e^3-*value* model, reflecting bundling decisions

Our four steps method provides a means to reason systematically about the selection of one service or another for a service bundle, eventually resulting in feasible service bundles that satisfy certain customer demands. When multiple feasible service bundles satisfy the same customer demands, it is important to be able to reason about differences between the bundles, to make a decision about one or more bundles, reflecting one or more business models to develop. Since we choose only a subset of the feasible bundles, our business analysis will have a much narrower scope than an analysis that takes all possible partners (and services) into consideration. The service ontology was applied to resolve the complexity problem of a business analysis in the energy sector by narrowing the scope of our primary business model. Consequently, significantly less effort had to be put into profitability assessment.

In our present study an energy supplier wishes to bundle electricity supply with other services, provided by a number of suppliers. The questions at hand are with which other services to bundle electricity supply, and whether the resulting business model(s) will be profitable. Past failures of similar initiatives show that these questions are far from trivial, and the competition in this sector requires a thorough analysis before a new business model is developed and a new service offering is mar-

keted. Even with a limited set of only ten services, the number of possible bundles gets exponentially high. We started with an initial business model, where a set of ten available services was identified, with which 1023 service bundles could theoretically be designed; assessing profitability for all service bundles would cost too much time. No mechanism was available for selecting bundles. By applying the service ontology in the energy domain, we managed to reduce the task complexity:

1. The number of service bundles for which profitability needs to be assessed was reduced by formalizing and applying dependencies between services, serving as rules for service bundling, or service configuration.

2. Knowledge on services was made available, to facilitate reasoning about a choice between feasible bundles.

3. Information on costs of and demand for services helps make a sound profitability assessment.

This knowledge is not available in the e^3-*value* ontology, where no guidelines are provided for bundling services. With a set of ten services, theoretically one would have to assess profitability for 1023 scenarios (1023 different sets of these ten services). The service ontology reduces this complexity. Step 2 of our method reduced the complexity to a maximum of several dozens solutions per scenario. Step 3 of our method further reduced the complexity to a few (typically two to five) service bundles per scenario. All other theoretically possible service bundles are irrelevant, and their profitability need not be assessed in step 4 of our method.

7.7.2 Ontology Development Perspective

Goal: use the service ontology to reduce complexity of business analyses
In this chapter we focus on the service offering perspective of our ontology, where we describe services as acts of exchanging economic values, and also as components for configuration. In the next chapter we focus on the service value perspective, showing (for a different study) how we ensure that the generated service bundles provide a good solution for customer demands.

Similarly, the study we describe here served us mainly for developing and validating the service offering perspective of the ontology. We modeled energy services using our *serviguration* ontology and software tool, and used our OBELIX software tool to generate service bundles based on criteria given by our business partner. Applying our ontology provides business developers with the required tools for performing a structured business analysis, reducing the complexity of the analysis by narrowing the number of possible business models that have to be analyzed.

Ontology validation: in exploratory real-world studies a 'good solution' is not defined a priori
From the perspective of TrønderEnergi AS, a study like this is aimed at designing and exploring new business models; TrønderEnergi AS does not know in advance which bundles shall be selected as sensible. Therefore, business developers cannot say in advance which service bundles they expect to be generated, such that we can validate the service model and service configuration algorithm by comparing results to a list of expected bundles. Instead, the theoretical validation of our ontology and related software tool works the other way around. We model services and generate service bundles. These are then presented to our business partner for assessment. In our case the generated service bundles were found sensible solutions by our business partner; our claim that service bundles can be configured by software when modeled using our service ontology proved to be correct in the energy study. The service ontology provided our partner with information and knowledge to continue the business analysis with profitability assessment of business models for offering service bundles.

Ontology development: real-world studies as a means to understand and generalize business logic
By modeling energy services and designing service bundles, we gained some major insights into the service offering perspective of our ontology. First, we used the study at hand to sharpen the definition of 'resources' in our service ontology, and to distinguish between a business value perspective on services and a process perspective. We model only resources that describe the value exchange between involved actors. Yet, as we have seen, some resources can belong to the business value perspective, as well as to the process perspective.

Second, energy services may have very complicated pricing models, allowing customers a high degree of flexibility in choosing a scheme that suits their needs best. This study was therefore very suitable for understanding how pricing models can be expressed as formulas, and how they can be modeled in the service ontology.

Third, due to the complex nature of this domain, many interdependencies exist between services. We refer not only to what we call 'service dependencies' in the ontology (determining which services can be combined into a bundle), but also to *how* one service influences another, assuming that they are bundled. For example, if customers have an energy control system next to their electricity supply, their electricity consumption is reduced by ten percent. The study at hand was used to develop the concept *conditional output* to model such constraints in the service ontology.

Ontology usage: discover the boundaries of an ontology
We used the study at hand to define an interface between our *serviguration* ontology and the *e*^{3}*-value* ontology for constructing business models, and implemented this interface in a software tool. By performing this analysis we learned the boundaries of both ontologies, and where they can fill in each other's gaps. The *e*^{3}*-value* ontology

provides the means for constructing business models, but it does not allow reasoning on how services should be bundled. The *serviguration* ontology, on the other hand, provides a means to reason about service bundling, but does not describe the resulting business model or allow for calculating profitability of a business model. We showed how both ontologies can be used together to perform a full analysis. The software-based interface between the ontologies allows for a computer-supported business analysis.

7.7.3 Concluding Remarks

An interesting feature of the energy sector is that this sector is growing horizontally, offering services which traditionally were not offered by this sector. Service bundles offered by energy utilities nowadays include non-energy related offerings, as a means to differentiate the energy product. This poses a challenge for business developers: criteria for including services in a bundle are sometimes still missing, and therefore an exploratory study is required, triggered by an understanding of customer needs as a key to designing new offerings. *Serviguration* focuses on customers as a starting point for designing service bundles. It is this specific characteristic of our service ontology that makes it so suitable for business analyses as presented in this chapter.

The study at hand presents evidence of the feasibility and usefulness of software-aided composition of service bundles. At the same time it also demonstrates the boundaries of automation, the places where human intervention is required. The method we present for business analyses can help business developers reduce the complexity of business analyses, and pinpoint good candidates for new service bundles. Yet, the choice for one or more service bundles – and related business models – has to be made by humans. When our method shows that several service bundles actually compete with each other because they provide a solution for a same or similar customer demand, human intervention is required to decide on a course of action. Decision criteria may not always be clear-cut. For example, one may choose to offer a service bundle with modest financial perspectives because it involves a reliable business partner, while an offering which may promise higher revenues necessarily involves working with a business partner that has already let you down in the past.

Chapter 8

Health Care and Welfare Services

***Note**: In this chapter we show how our service ontology is used in the health sector to design and offer service bundles, mainly for people with dementia and their (in)formal carers. An early position paper concerning this study was published in Medical and Care Compunetics 2, Volume 114 Studies in Health Technology and Informatics (Dröes, Meiland, Doruff, Varodi, Akkermans, Baida, Faber, Haaker, Moelaert, Kartseva & Tan 2005).*

Early in this thesis we claimed that our service ontology can be used for business analyses and for developing websites through which suppliers can jointly offer their services to customers. The first usage was exemplified by a study in the energy sector in Chapter 7. In the current chapter we exemplify the second usage of our service ontology. We present a study that is part of the large scale FrUX research project in the health sector and police in the Netherlands.[1] We show how our service ontology is used to design customer-tailored cross organizational service bundles for people with dementia and for their (in)formal carers. Our service ontology and this study are currently being used as the fundaments for developing software for a website through which these service bundles can be offered to customers. The study at hand provides evidence of the usefulness of our service modeling approach.

8.1 Domain: Dementia Care

E-Service bundles may play an important role in the field of care and welfare for elderly persons with dementia and their carers, as a solution for a number of problems that this field faces now and will face in the near future. Dementia is a disease that

[1]FrUX (Freeband User eXperience, http://frux.freeband.nl) is partly funded by the Dutch Ministry of Economic Affairs.

mainly affects older people (Health Council of the Netherlands 2002). Nearly 1% of the persons of 65 suffer from the disease, and this figure rises to 40% in people of 90 years and older. Currently, there are over 175,000 persons with dementia in the Netherlands (Health Council of the Netherlands 2002). Dementia is not a disease in its own right, but the name given for a combination of symptoms. There are different diseases in which dementia can occur and up till now there is no effective treatment available. Dementia is a progressive disorder in which the person becomes increasingly dependent on others for his daily functioning. Important features of dementia are disorders of the memory, speech, thinking, perception, reasoning and performing activities. Often there are also changes in personality, behavior and psychological functioning, such as depressive symptoms, apathy and aggression. These changes do not only affect the person with dementia himself, but also the persons in their environment. About 65% of the persons with dementia live in the community and are cared for by informal carers, such as spouses, children, and other relatives. Persons with dementia and their informal carers wish to stay in the community as long as possible and this is in line with the current Dutch policy aims. However, taking care of a person with dementia is recognized as a burdensome task (Timmermans 2003). Many informal carers experience negative physical, psychological and social consequences (Burns 2000, Coope et al. 1995, Eagles et al. 1987, Meiland et al. 2001, Pot 1996, de Vugt et al. 2003, Zarit et al. 1980, Dröes et al. 2004) that besides the behavioral problem of the person with dementia, are important determinants for nursing home admission (Greene et al. 1980, Teri 1997, Braekhus et al. 1998, Kaufer et al. 1998, Mirakhur et al. 2004, Aalten 2004, de Vugt 2004).

It is therefore important that health care and welfare services not only focus on care and support for the persons with dementia but also for their informal carers. Providing informal carers of people with dementia with supporting welfare services will increase their ability to cope with the heavy daily burden, and will allow them to continue nursing the patients longer, thereby postponing the nursing home admission of patients. It will also help prevent them from getting overburdened, eventually requiring health care services themselves. Example health care services are illness diagnosis and treatment. Welfare services include help with small tasks at home (e.g., hanging a curtain, repairing an electricity socket), telephone help line, assistance with financial administration and more. These health care and welfare services help persons with dementia and their informal carers cope with the consequences of dementia. Also, these services ease the life of informal carers, so that they can more easily support their ill relatives and friends.

Because of the expected growth of the group of elderly persons with dementia, and chronically ill people in general, and as a consequence of the enormous increase of informal carers in the coming decades, it is necessary to increase our specificity and efficiency in delivering services. This means more specific, individualized and dynamic service bundling, tailored precisely to the needs expressed by the client-

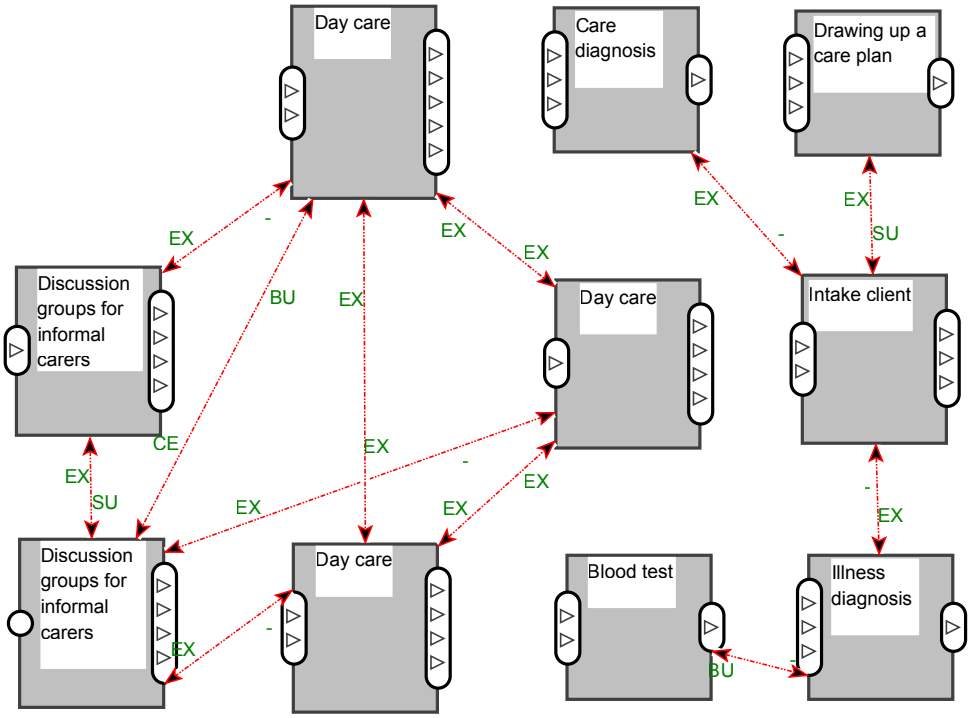

Figure 8.1: Example health care and welfare services for people with dementia and their informal carers. Note how a same service (e.g., day care) is modeled multiple times because it is offered by a number of service providers, and each requires and/or provides other resources.

system (patient and carers) (Dröes et al. 2005).

Currently, several problems exist in the field of care and welfare for persons with dementia and their carers. These problems include the variation, fragmentation and continual changing of care and welfare services in a region, both public and private. Clients and referrers cannot see the wood for the trees anymore and therefore tend not to utilize the broad spectrum of available services. Possible consequences are: lacking the specific care and support one needs, unsafe situations, social isolation of patients and frustration, overburden and illness of carers. Thus, the need for a more transparent, easy accessible and integrated offer of health care and welfare services is growing.

In our study we used a dataset of 38 health care and welfare services, supplied by a variety of service providers. A subset thereof is visualized in Figure 8.1 and described in detail in Table 8.3 (more services are described in examples throughout this chapter). Theoretically, 274,877,906,943 ($2^{38} - 1$) different service bundles (of one or more services each) can be generated from this dataset. The large number

emphasizes why clients cannot see the wood for the trees. And yet, in reality there exist more than 38 services. The service ontology is required to cope with this large dataset, and design only service bundles that are based on sound business logic, from a customer perspective and from a supplier perspective.

Another problem is the generally recognized need to create a continuum of flexible care and welfare bundles in every region in the Netherlands, that dynamically meet the care needs and wishes of individual persons with dementia and their informal carers in the different stages of the disease. To understand what gaps exist in the present offering, insight into the care needs and wishes of this client group and their informal carers as well as an up-to-date overview of regional (and national) services are prerequisites. Recently, a first step was made in collecting this type of information: a National Dementia Program (NDP) was developed which describes needs of the target group and examples of potential care offerings (Meerveld et al. 2004). This NDP was produced by order of the Ministry of Public Health, Welfare and Sports as a response on the latest advice on care and research in dementia by the Health Council of the Netherlands (Health Council of the Netherlands 2002). This overview of needs and care solutions in the NDP offers a good starting point for understanding the needs of persons with dementia and their carers in the community, and to fill in gaps in the care and welfare offerings.

FrUX, a project within the Dutch *Freeband* program, addresses these problems in dementia care. Our study uses the NDP as a starting point to inventory needs and support offerings. We do this by means of a literature study on subjective needs of the target group and by a field study with standardized and open interviews with persons with dementia, their informal carers and professional carers. Furthermore, the current care and welfare offerings are being inventoried in two regions in the Netherlands (Amsterdam-Zuid and Nijmegen). A gap analysis will be performed between needs and current service offerings, to trace ICT opportunities that could contribute to an improvement of the care and welfare service offerings for persons with dementia and their (in)formal carers. The ICT opportunities we search for are context-aware services that support interaction between people in dynamic social contexts (van Eijk et al. 2004). Context-awareness means that the service is aware of people's context in general and more specifically their location, social context and activities.

While the study on customer needs is still ongoing, we use the NDP as a starting point for identifying ICT opportunities for improving health care and welfare service offerings. One of the ICT opportunities that could be developed and that could start to address the problems mentioned above, is a Dynamic Interactive Social Chart for Dementia Care (DEM-DISC), a service that will offer service bundles that are adjusted to the context(s) of people. Furthermore, it will support interaction by enabling people to communicate, exchange information (about themselves) or collaborate. DEM-DISC will be an interactive regional social chart that is accessible via

the Internet and that, besides providing information on care and welfare services to persons with dementia and their informal carers, is also able to respond dynamically to their individual needs with customized care and support information.

8.2 Context: How One Customer Differs from Another

8.2.1 Reasoning about Context Using the Service Ontology

An individual solution per customer requires that DEM-DISC takes the context of every customer into consideration. This will take place in two phases in DEM-DISC. In the first phase we design service bundles that fit a group of customers who share certain characteristics (e.g., age group, location), and in the second phase we design service bundles that take customer profiles into consideration, and interact with customers to obtain the required information about them, so that personalized service bundles can be generated by the software. In both cases, the trigger for the bundling process will be knowledge about the needs of a customer or customer group.

This is facilitated by the service ontology using what we call context information. *Context* refers to the physical and social situation of (in our case) customers of DEM-DISC. Examples are time, location and age; two patients may have the same demand, and yet require different treatments to satisfy this demand because of their different ages (people beyond a certain age are considered to be too weak for certain treatments). Hence, the relation between customer needs (as captured by need hierarchies) and available services, and the choice of services to include in a bundle depend on the context of a given customer, or a customer group. Service bundles should be designed for customers who have certain needs, and are in a certain context. A context has a name and a value, for example ‘name: age, value: 65’ or ‘name: customer type, value: informal carer’. Naturally, multiple contexts may be valid simultaneously.

Context information can be handled in three ways using the service ontology:

1. Some context information can be considered as assumptions that narrow the scope of the information system to be developed. Using such global assumptions, this information need not be modeled separately for every element in the system.
   Example: DEM-DISC is targeted at people with dementia; consequently when we say “patient” we mean a dementia patient, and not a cancer patient.

2. Some context information describes the conditions under which a given benefit (resource) can satisfy a given customer demand (we refer to this as “context on the resource level”). This information is modeled in production rules between demands and resources (a given demand triggers the choice of different

resources, when different contexts apply).
Example: People with dementia and their informal carers may have a demand for companions contact. Both will seek for social and/or emotional support, modeled in the service ontology as experience resources. Yet, emotional/social support for people with dementia is different from emotional/social support for informal carers; these are different resources. Consequently, the demand for companions contact requires different resources, based on the customer type (modeled as context information).

3. Some context information describes the conditions under which a whole service element qualifies (or does not qualify) as a solution, when multiple service elements deliver the same benefits, and yet not all of them are always a proper solution (we refer to this as "context on the service level"). This is supported by the relation "service element has context".
Example: A service for renting appliances (e.g., a wheelchair) is provided only to customers who live in a certain region. We model this geographical restriction as context information, related to the appliances rental service.

8.2.2 Context Information in Dementia Care

We modeled the following context information in the dementia study:

1. Two customers interested in (illness) diagnosis will be offered different resources based on their age, resulting in a selection of different services (every service provides different resources). This is due to the assumption that people older than 78 are not healthy enough to undergo a certain physical check.

2. Services of the supplier "Thuiszorg" (Home Care) are provided only to customers who live at home, and not to people who live in institutions. Since DEM-DISC is designed for customers who still live at home, this context information is a global assumption, and need not be modeled for the services of this service provider.

3. Some services of a district post for elderly ("Wijkpost voor ouderen", in Dutch) are provided only to people aged 55/65 (depending on the service) or older, with a low income. If the customers suffer from dementia, the restriction for low income is dropped. This is modeled by restricting the services of this service provider to customers for whom these context elements are valid.

4. The location where the service is provided is a constraint on offering services to a customer. One may decide to offer services only to customers within a certain geographic region (note that the location where the service is provided is not necessarily the same as the address of the service provider).

Table 8.1: Context information in dementia services

Context information	*Type of context information*
Age	Context on the resource level
Living at home vs. living in an institute	Global assumption
Age and income	Context on the service level
Place of providing the service	Context on the service level
Place of living	Context on the service level
Customer type/group	Context on the resource level
Relevant disease/illness	Context on the service level

5. Certain services for hiring appliances (e.g., a wheelchair) for a predefined period are provided only to customers who live in a specific region.

6. When comparing the demands of various patients and informal carers, one can see that they may have a same demand, for example the demand for companions contact (in Dutch: "lotgenotencontact"). This demand may be (partly) satisfied by a service that provides emotional support. Several services may offer emotional support, but some of them are targeted at patients (hence an informal carer will not obtain the desired support upon consuming this service), and others are targeted at informal carers (hence a patient will not obtain the desired support if he consumes this service). In other words, the same demand requires a different emotional support resource for different customer types.

7. An online chat service for patients who suffer from Pick's disease will be offered only to people who actually suffer from Pick's disease or to their carers.

While the above list is not complete (as we have modeled only information which is relevant to the demands and services in our study), it is representative. Table 8.1 presents an analysis of this context information. Embedding the concept *context* in the service ontology enables us to reason about the suitability of solutions (service bundles) for given requirements (demands of a customer or customer group).

8.3 Modeling Supply and Demand for Dementia Care

We modeled domain knowledge about dementia care using the concepts of our service ontology, including the supplier perspective, the customer perspective and a transformation between the two. The modeling effort was performed iteratively in collaboration by domain experts and knowledge modeling experts.

8.3.1 Demand Side (Customer) Perspective

For the customer perspective our starting point is the National Dementia Program NDP (Meerveld et al. 2004), where 14 problem areas are defined, of which we selected three. These were the areas "what is the problem and what can help?", "having to face everything on your own" and "cannot cope anymore". For every such problem area we defined a hierarchy of needs, wants and demands for two customer groups: people with dementia and their informal carers (major parts of the two need hierarchies overlap). The main part of the need hierarchy for informal carers is shown in Table 8.2. Many of the demands were also specified using qualitative descriptors as 'personal', 'in group', 'online', 'in writing', 'oral' and more, referred to as service properties in the service ontology. Such descriptors help make a demand concrete enough so that a solution can be found for it. As can be seen from the table, the same demand may be related to a number of wants and needs.

Table 8.2: Examples of needs, wants and demands of informal carers of people with dementia (this table is split across pages).

Customer Needs	*Customer Wants*	*Customer Demands*
Know what is the problem and what can help?	Know whether this is an illness	Illness diagnosis
	Know more about the illness	Information about the illness (general or personal information; oral or in writing)
	Know what are the consequences for the daily life with a person with dementia	Information about the consequences for the daily life and household activities of a person with dementia (general or personal information; oral or in writing)

continued on next page

continued from previous page

Customer Needs	*Customer Wants*	*Customer Demands*
		Information about the consequences for the psychological functioning of a person with dementia (general or personal information; oral or in writing)
	Know what are the consequences for the daily life of an informal carer	Information about the practical consequences for the daily life of an informal carer
		Information about the emotional consequences for the daily life of an informal carer
		Information about the social consequences for the daily life of an informal carer
		Companions contact for informal carers of people with dementia
Having to face everything on your own	Help with practicalities in the daily life with a patient	Education concerning practicalities in daily life with a patient
		Assistance with housekeeping
		Assistance with financial administration
		Food catering
		Wheelchair rental
	Social support	Companions contact for informal carers of people with dementia (via the Internet or in the physical world; one-on-one or in groups)
		Discussion group for informal carers of people with dementia (via the Internet or in the physical world)
	Support in coping with the changing behavior of a patient	Discussion group concerning how to cope with the changing behavior of a patient
		Information concerning how to cope with the changing behavior of a patient

continued on next page

continued from previous page		
Customer Needs	*Customer Wants*	*Customer Demands*
Cannot cope anymore	Practical support	Food catering
		Assistance with housekeeping
		Assistance with financial administration
		Wheelchair rental
		Admission into a home for the elderly
		Admission into a nursing home
		Supervision
	Social support in case of overburdening	Assistance in building up a social network
	Emotional support in case of overburdening	Companions contact for informal carers of people with dementia (via the Internet or in the physical world; one-on-one or in groups)
		Discussion group for informal carers of people with dementia
	Temporary relief from care	Temporary admission into an institution
		Day care (with social or therapeutical character)

8.3.2 Supply Side Perspective

We modeled 38 services that are related to the above demands. Ten services are visualized in Figure 8.1 and described in detail in Table 8.3. A few remarks are in place concerning the services presented in Table 8.3:

- Services of a GP (general practitioner) require an input of type human resource: GP time slot. We model human resources only when they are not inherent to the service. A patient who undergoes an operation pays a fixed amount for the operation. The price of a GP's service, on the other hand, depends on the number of time slots that it requires (if a treatment requires ten visits to the GP, the patient will pay for ten visits). Therefore we model the GP as a human resource.

- We model information resources only when they present economic value for some actor. For example, several services require a dementia diagnosis and a

care diagnosis. Various providers provide these, for a fee. Therefore dementia diagnosis and care diagnosis are modeled are resources.

- Services may deliver similar resources, with differing quality levels. For example, discussion groups are offered by meeting centers and by the GGZ,[2] providing the same resources, but while the GGZ provides expertise in psychiatry, this is not always the case for discussion groups in meeting centers. Quality is modeled by service properties that describe input- and outcome resources (these are omitted from the table for readability).

We modeled services that are supplied by a variety of health care providers, making the study realistic and interesting: the majority of all possible bundles is cross-organizational. The services we modeled include (illness) diagnosis, care diagnosis, drawing up a care plan, blood test, medication advice, discussion groups for informal carers, help with small tasks at home, telephone help line, assistance with financial administration and more. For each service we modeled the exchange of values (benefits: service inputs and service outcomes) between customers and suppliers. Some services were modeled several times, because the services are provided by various service providers, each of which provides *different benefits*, although the *functionality* is the same. Functionality alone is not enough to determine which of the services can best suit a specific customer. Analyzing the different benefits of these services provides the required knowledge to reason about the suitability of a service for a given customer. This observation supports our approach, in which the benefits of a service, rather than the functionality thereof, are used to match available services with customer demands.

Table 8.3: Health care and welfare services described by their input- and outcome resources (this table is split across pages).

Service name and supplier	*Service inputs*	*Service outcomes*
Name: discussion group for informal carers Supplier: meeting center	none	emotional support for informal carers (experience resource); social support for informal carers (experience resource); practical advice for informal carers (information resource); coping advice for informal carers (information resource)
		continued on next page

[2]The GGZ is a mental health care organization for the prevention and treatment of psychological and psychiatric symptoms and diseases.

continued from previous page

Service name and supplier	*Service inputs*	*Service outcomes*
Name: discussion group for informal carers Supplier: GGZ	fee (monetary resource)	emotional support for informal carers (experience resource); social support for informal carers (experience resource); practical advice for informal carers (information resource); coping advice for informal carers (information resource)
Name: day care Supplier: home for the elderly	formal indication (capability resource) (the indication system determines the entitlement of patients for professional care)	social contacts for people with dementia (experience resource); supportive assistance for disabilities of persons with dementia (experience resource); supervision (experience resource); respite for informal carers (capability resource)
Name: day care Supplier: nursing home	fee (monetary resource) ; formal indication (capability resource)	social contacts for people with dementia (experience resource); supervision (experience resource); respite for informal carers (capability resource); reactivation (state-change resource)
Name: day care Supplier: meeting center	fee (monetary resource) ; formal indication (capability resource)	supervision (experience resource); respite for informal carers (capability resource); reactivation (state-change resource); emotional support for person with dementia (experience resource); social support for person with dementia (experience resource)

continued on next page

continued from previous page

Service name and supplier	*Service inputs*	*Service outcomes*
Name: intake client Supplier: GGZ	fee (monetary resource) ; (doctor's) referral letter (monetary resource)	dementia diagnosis (information resource); care diagnosis (information resource); care plan (information resource)
Name: illness diagnosis Supplier: GP (general practitioner)	fee (monetary resource) ; blood test report (information resource) ; GP time slot (human resource resource)	dementia diagnosis (information resource)
Name: blood test Supplier: lab	fee (monetary resource) ; (doctor's) referral letter (monetary resource)	blood test report (information resource)
Name: care diagnosis Supplier: GP (general practitioner)	fee (monetary resource) ; dementia diagnosis (information resource) ; GP time slot (human resource resource)	care diagnosis (information resource)
Name: drawing up a care plan Supplier: outpatient memory clinic	fee (monetary resource) ; dementia diagnosis (information resource) ; care diagnosis (information resource)	care plan (information resource)

An important part of modeling the supply-side perspective is defining service dependencies, business rules for composing bundles out of elementary services. Example business rules are:

- Meeting centers provide services for people with dementia, as well as services for informal carers of these patients. The core business of meeting centers is providing *combination therapy*: services for people with dementia *in combination* with services for informal carers of these patients. A meeting center does not provide services to persons with dementia, if their informal carers do not participate in the therapy, and vice versa. Thus, any service bundle that includes a service of a meeting center for informal carers will also include a service for people with dementia.[3] We model this using the *bundled* service dependency.

[3]One exception exists for the above rule: the service "education concerning dementia related problems" of meeting centers (informative meetings) is available for all interested people.

- Several service providers offer day care. Some of the outcomes they provide are similar, but others differ. Yet, their main functionality is the same, and a care expert will not offer more than one day care service to a patient. In other words, these services exclude each other. We model this using the *excluding* service dependency.

- A number of service providers provide the service of drawing up a care plan. The GGZ offers a *total package* called "intake client", which in fact includes drawing up a care plan, as well as other services. Putting it differently, the outcomes of the service "drawing up a care plan" are a subset of the outcomes of the service "intake client" of the GGZ. Therefore "intake client" is a substitute for "drawing up a care plan" (but not vice versa). We model this using the *substitute* service dependency.

- CIZ ("Centrum Indicatiestelling Zorg", in Dutch), is a governmental agency that determines the entitlement of patients for professional care, described in so-called 'functions'. This entitlement is called *formal indication*. A number of services require that patients have an indication. The service "indication" of CIZ (not modeled in Table 8.3) is thus a supporting service for the services that require an indication. All these services have a core/supporting service dependency with "indication".

- A distinction is made between services that are provided as long as the person with dementia lives at home, and services that involve admission into a nursing home, to a home for the elderly or to a hospital. Services of both groups exclude each other, because a customer cannot receive care at home and be admitted into an institution at the same time. We model this using the *excluding* service dependency.

A representative subset of service dependencies is presented in Table 8.4.

8.3.3 Transformation Between Perspectives

Next, we modeled production rules to define which resources (outcomes of services) satisfy which demands, using the four production rules presented in Chapter 5: *selection* (resource Y must be selected in case of demand X), *rejection* (resource Y must *not* be selected in case of demand X), *positively influenced by* (resource Y may be selected as it contributes positively to satisfying demand X) and *negatively influenced by* (the availability of resource Y in a service bundle may have a negative effect on satisfying demand X, but the demand can still be satisfied, although not optimally). Production rules can describe a relation between a demand and a resource, independent of the quality descriptors of the demand and the resource, or a relation that holds

Table 8.4: Examples of service dependencies in dementia care

Service dependency	*Dependee: service name (service provider)*	*Dependent: service name (service provider)*
Substitute	Discussion group for informal carers (GGZ)	Discussion group for informal carers (meeting center)
Excluding	Day care (home for the elderly)	Temporary admission (nursing home)
Core/supporting	Day care (home for the elderly)	Indication (CIZ)
Excluding	Day care (home for the elderly)	Day care (meeting center)
Core/enhancing	Day care (meeting center)	Discussion group for informal carers (meeting center)
Excluding	Day care (meeting center)	Temporary admission (nursing home)
Bundled	Treatment (outpatient memory clinic)	(Illness) diagnosis (outpatient memory clinic)
Substitute	(Illness) diagnosis (GP)	(Illness) diagnosis (outpatient memory clinic)
Substitute	Drawing up a care plan (outpatient memory clinic)	Intake client (GGZ)
Excluding	Intake client (GGZ)	Drawing up a care plan (outpatient memory clinic)
Optional bundle	Medication counseling (GP)	(Illness) diagnosis (outpatient memory clinic)
Core/supporting	Care diagnosis (outpatient memory clinic)	(Illness) diagnosis (outpatient memory clinic)
Bundled	(Illness) diagnosis (outpatient memory clinic)	Care diagnosis (outpatient memory clinic)

only when the resource and/or demand are described with specific quality descriptors. We use the following terminology in describing production rules:

- D1 annotates a demand; R1 annotates a resource; Q1, Q2, Q3... annotate service properties of demands and resources.
- D1Q1 annotates a demand that is specified by a service property. Similarly, R1Q1 annotates a resource that is specified by a service property.
- When we say "parents" we refer to a pair of a demand and a resource where both the demand and the resource are not specified by service properties, i.e., (D1, R1).
- When we say "siblings" we refer to a pair where either the demand, or the resource, or both are specified by service properties, i.e., (D1, R1Q2), (D1Q1, R1) or (D1Q1, R1Q2). These three pairs are the siblings of (D1, R1). This can be generalized to a situation where the demand and/or resource are specified by multiple service properties. In this case, also (D1, R1Q2Q3), (D1Q4, R1Q2Q3) and so forth are siblings of (D1, R1).

Examples of production rules from the dementia study are given below, and the business logic behind them is explained. We describe production rules similarly to how we described them in Section 5.3, where we explained how to reason with production rules.

Example 1:
Demand D1: Companions contact for informal carers of people with dementia
Demand D1Q1: Companions contact for informal carers of people with dementia, with service property Q1: contact type: via the Internet
Resource R1: Knowledge concerning care and support offer in the region for persons with dementia and their carers (information resource)
Resource R1Q2: Knowledge concerning care and support offer in the region for persons with dementia and their carers, with service property Q2: information type: oral
Production rules: POS(D1, R1), REJ(D1Q1, R1Q2)
Explanation: Domain experts recognize that when informal carers require companion contact, they need more than social and/or emotional support; they also lack knowledge that they hope to gain through companions contact. Therefore if a service bundle offers this knowledge, it will be a better solution than a service bundle that does not offer this knowledge. This is modeled by the parents' production rule POS(D1, R1). Several services may offer the same information resource R1 with different properties: some offer it in writing (e.g., via brochures), and others orally (e.g., through counseling). If informal carers specifically specify that they wish the companions contact to be online, any service that offers the same information resource orally is

not a good solution. We model this with the production rule REJ(D1Q1, R1Q2). To summarize, if a customer's demand is "companions contact for informal carers of people with dementia, via the Internet", the *serviguration* algorithm will search for services that provide the resource "knowledge concerning care and support offer in the region for persons with dementia and their carers", but any service that provides this resource orally will be disqualified.

Example 2:
Demand D1: Assistance through consultations
Resource R1: Social support for informal carers (experience resource)
Context: Customer type: informal carers of people with dementia
Production rules: POS(D1, R1)
Explanation: For the demand "assistance through consultations" to be satisfied it is not required that social support is provided. Yet, domain experts recognize that such support influences the demand satisfaction positively, as we model with a POS production rule. Note that this production rule is context dependent: it is triggered only when the customer is an informal carer. If the customer is a patient, a similar production rule will be triggered, with a resource R2: "social support for people with dementia". Although R1 and R2 offer the same functionality, social support for informal carers differs from social support for people with dementia (and they may very well be delivered by different services, although also a single service may deliver both). Consequently, the two are modeled as two different (experience) resources, and each is triggered in a different case.

Example 3:
Demand D1: Companions contact for people with dementia
Demand D1Q1: Companions contact for people with dementia, with service property Q1: accessibility: low barrier
Resource R1: Knowledge concerning dementia related problems (information resource)
Resource R1Q2: Knowledge concerning dementia related problems, with service property Q2: expertise: psychiatry
Production rules: REJ(D1Q1, R1Q2)
Explanation: As we did not model a production rule between the parents, when a customer seeks for companions contact for people with dementia, we will not actively search for services that provide the resource "knowledge concerning dementia related problems". And yet if a generated bundle provides this resource, it will not be disqualified (because there is no disqualifying production rule: REJ or NEG). Only when we know that a customer is interested in low barrier solutions, will we disqualify services that offer a psychiatric expertise, because they typically present a high accessibility barrier for customers.

Example 4:
Demand D1: Temporary admission into an institution
Resource R1: Respite for informal carers (capability resource)
Resource R1Q1: Respite for informal carers, with service property Q1: number of hours per day: 24
Production rules: SEL(D1, R1Q1)
Explanation: Many services may offer respite for informal carers; some of them relieve informal carers from their care tasks for a few hours per day only, while others offer the same respite benefit for the whole day. All these services offer a "respite" benefit, but not all of them will have the property "number of hours/day: 24". As admission into an institution implies 24 hours/day care, when customers ask for temporary admission, any service bundle must include a service that provides a 24 hours/day respite resource.

Example 5:
Demand D1: Discussion group for informal carers
Demand D1Q1: Discussion group for informal carers, with service property Q1: accessibility: low barrier
Resource R1: Social support for informal carers (experience resource)
Resource R1Q2: Social support for informal carers, with service property Q2: accessibility: low barrier
Production rules: POS(D1, R1), POS(D1Q1, R1Q2)
Explanation: When informal carers ask for a discussion group, they often seek social support. Therefore we model the POS relation between the parents demand and resource as specified in this example. Various services may offer various "social support for informal carers" resources with different properties. When customers specify that they are looking for a solution with a low accessibility barrier, we will search for solutions where the mentioned resource is described by this property. The social support resource is mostly described also by other service properties, in addition to the accessibility property. Yet, if the demand specifies only property Q1 (low accessibility barrier), no other properties of the resource will be taken into consideration, because the only production rule involving D1Q1 concerns the accessibility property of the resource. For example, the resource "social support for informal carers" is also described by the service property Q3: "expertise: psychiatry". But because no production rule is modeled between D1Q1 and R1Q3, the expertise property will not be taken into consideration in the search for a solution.

8.3.4 Serviguration: How We Design Service Bundles

Knowledge about customer demands, available services, service dependencies, context information and production rules enables us to define service bundling requirements and to configure services into service bundles. We termed this process *servi-*

guration , and explained it in Chapter 3 (see Section 3.5 and Figure 3.13). Customer demands are used as a trigger for *Serviguration.* In short, based on a given set of customer demands and acceptable sacrifices (and possibly customer context information), we use production rules to derive a set of resources that can satisfy the given demands. Services that offer these resources form initial solutions. Service dependencies determine which other services may, must or must not be added to these initial solutions, eventually resulting in service bundles that we present to users as solutions.

Take for example a customers' demand for advice concerning possibilities of care in the region for coping with the consequences of dementia. When this demand is set, a number of production rules are triggered, indicating that a variety of resources can contribute to satisfying this demand. One such resource is "care plan", an information resource. Therefore the *serviguration* algorithm searches for services that provide a care plan. This example is of special interest, because a number of service providers – e.g., an outpatient memory clinic and the GGZ – offer services that deliver a care plan. However, drawing up a care plan first requires an illness diagnosis and a care diagnosis, so in fact the requirement for a care plan implies that also services for illness diagnosis and care diagnosis should be part of a bundle that provides a care plan.

Several service providers provide services that result in the resources "illness diagnosis" and "care diagnosis", for a fee (which may be reimbursed by a medical insurance). These include a GP, an outpatient memory clinic and the GGZ.

Theoretically, we could design a service bundle where the resources "illness diagnosis" and "care diagnosis" are provided by a GP, and the resource "care plan" is provided by an outpatient memory clinic. This, however, is not possible, because the outpatient memory clinic requires a specialist's care diagnosis, while a GP is not a specialist. This restriction is modeled in the service properties of the input- and outcome resources of the various services. For example, a GP and an outpatient memory clinic both provide services that result in the resource "care diagnosis". However, the two outcomes of the two services have different service properties that describe whether the service provider is a specialist or not.

A possible service bundle that provides the required "care plan" is a "total package" service bundle, provided by the outpatient memory clinic, including illness diagnosis, care diagnosis and drawing up a care plan, as visualized in Figure 8.2.

We will now analyze another example in more detail, and emphasize the various steps in the *serviguration* process.

***Serviguration* input: customer requirement and context.**
An interesting example involves a very commonly heard demand: companions contact for informal carers of people with dementia. We assume that the demand is specified by two service properties: (1) contact type: in real world (as opposed to contact

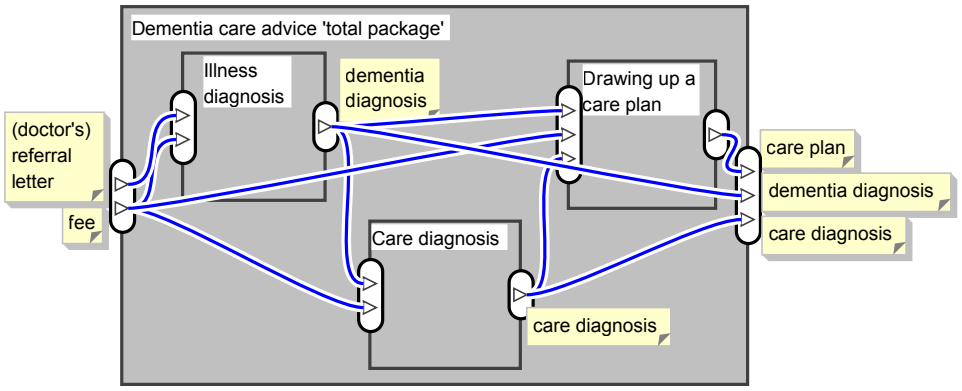

Figure 8.2: Service bundle offered by an outpatient memory clinic

via the Internet), and (2) personalization level: in groups. We further assume some context information: the customer is interested in a solution for an informal carer only, not involving the person with dementia. We model this by context information; the customer type is 'informal carer', rather than "informal carer" *and* "patient".

Production rules: a demand is satisfied by benefits (resources).
'Companions contact' is in fact a broad term; it may include social support, emotional support, knowledge and advice. Therefore this demand triggers a large number of production rules, involving, among others, the following resources:

- Emotional support for informal carers (experience resource)
- Social support for informal carers (experience resource)
- Practical advice for informal carers (information resource)
- Coping advice for informal carers (information resource)
- Knowledge concerning care and support offer in the region for persons with dementia and their carers (information resource)

Production rules are used for defining customer requirements in supplier terminology (resources). They result in a set of resources that must, may or may not be part of any solution bundle. In our case, no *selection* production rule was triggered. Therefore, none of the above five resources *must* be part of every solution. Similarly, no *rejection* production rule was triggered. Therefore, no resource is per definition *excluded* from solution bundles. The production rules involving the above five resources were of the type *positively influenced by*, implying that the availability of any of these resources (independently) contributes to satisfying the demand, but the demand can also be met without that resource. The given customer demand can best be satisfied by a service

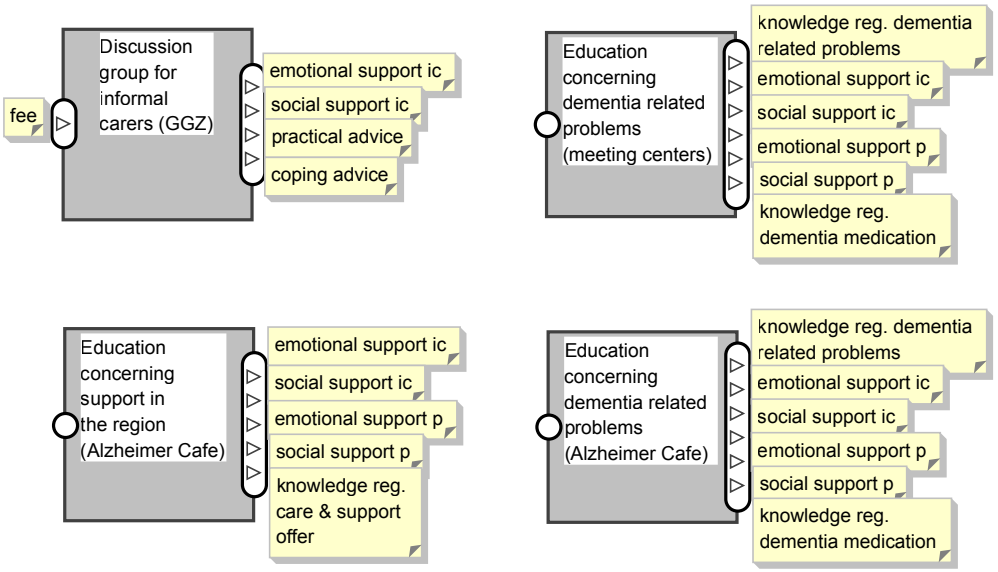

Figure 8.3: Services that can be offered when informal carers ask for companions contact

bundle that provides all five resources. And yet, also service bundles that provide only a subset of these five resources are suitable solutions.

Initial solution bundles include services that provide at least some of the desired resources.
Seven different service bundles provide a solution for the given demand. They are combinations of four services, described hereunder and visualized in Figure 8.3. As explained before, and can be seen in the figure, we distinguish between emotional/social support for informal carers (denoted "emotional/social support ic" in the figure) and emotional/social support for people with dementia (denoted "emotional/social support p" in the figure). Note how three of the four services are offered for free, while the service of the GGZ requires a fee (which may be reimbursed by a medical insurance). The four services are:

- Discussion group for informal carers, provided by the GGZ (this service is described in Table 8.3)
- Education concerning dementia related problems, provided by meeting centers
- A combination of education concerning dementia related problems and education concerning support in the region, both provided by Alzheimer Cafés[4]

[4]An Alzheimer Café organizes monthly informal meetings for people with dementia, their part-

A remark is in place here. Due to the complexity of real-world situations, in studies like ours it is customary to model *part* of a domain, and to consider this part as the study's Universe of Discourse. We modeled 38 services, but in reality more services exist. For example, meeting centers actually provide education concerning support in the region, as do Alzheimer Cafés. However, we only modeled the service "education concerning support in the region" of Alzheimer Cafés, and not that of meeting centers. Because this service was not part of our model, the current discussion assumes that such a service does not exist.

Final solution bundles are achieved by applying service dependencies to initial solutions.
The two different services provided by Alzheimer Cafés are provisioned only in combination, as one package. We model this by two 'bundled' service dependencies between them. Any service bundle that includes one of them will also include the other.

***Serviguration* output: possible solution service bundles.**
None of the four services provides all five resources as specified in our requirement. Yet, as explained before, none of these resources is a hard requirement, so also solutions that provide only a subset of the five resources are suitable solutions. Only one of the services (a discussion group of the GGZ) provides the resources 'practical advice for informal carers' and 'coping advice for informal carers'. Similarly, only one of the services (education concerning support in the region, provided by Alzheimer Cafés) provides the resource "knowledge concerning care and support offer in the region for persons with dementia and their carers". Therefore, a bundle that provides all five resources will necessarily include both these services. Furthermore, because the two services provided by Alzheimer Cafés are provisioned in combination, a service bundle that provides all five resources would have to include at least three – and not just two – services, as depicted in Figure 8.4. One of these services, namely the service of the GGZ, requires a fee. The others are free of charge.

If we wish to keep the solution service bundles free of charge, the bundle will not provide two of the five desired resources (practical advice and coping advice), because they are only provided by a service of the GGZ, which is not free of charge. A free of charge service bundle would therefore include at least the combination of the two different Alzheimer Cafés' services, as depicted in Figure 8.5. A more elaborated solution – and still free of charge – would include also the service of meeting centers. Although this service bundle, depicted in Figure 8.6, does not present new benefits (service outcomes) that are not present in the more limited bundle (all outcomes of the new service are already present in the bundle), it provides *more* of the same outcomes. Customers may value the extra support and information that they will obtain at a meeting center.

---

ners, family members, carers and other interested people. These meetings are organized by regional organizations, and supported by the *Alzheimer Nederland* foundation.

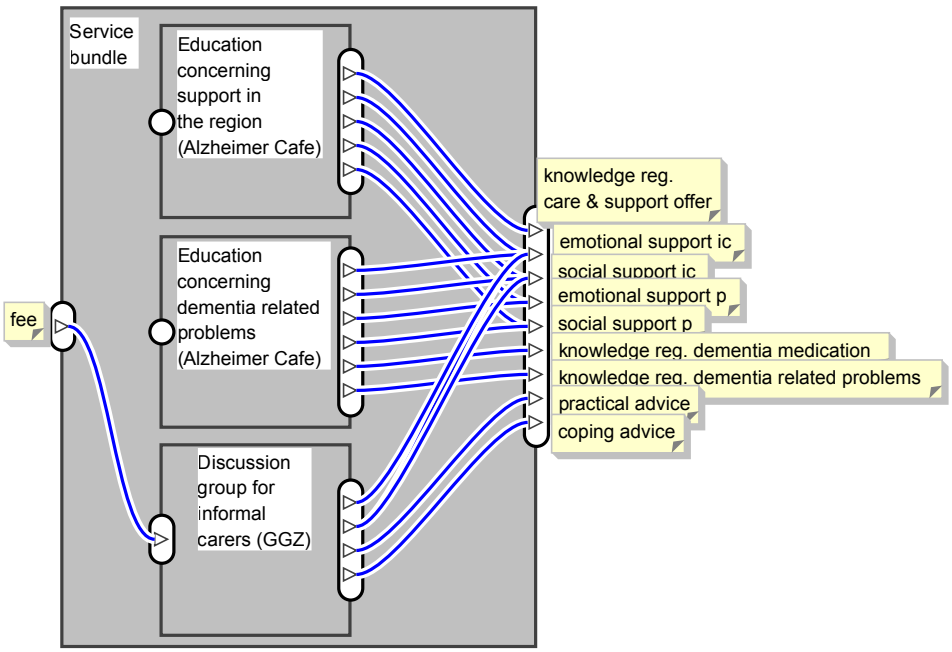

Figure 8.4: A service bundle for informal carers who ask for companions contact

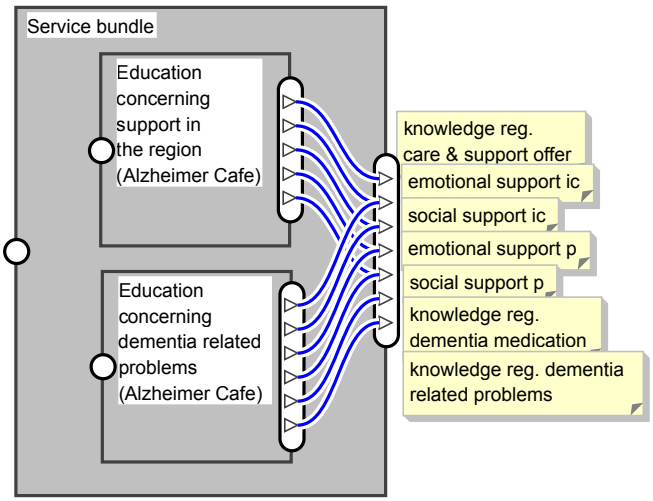

Figure 8.5: A free-of-charge service bundle for informal carers who ask for companions contact

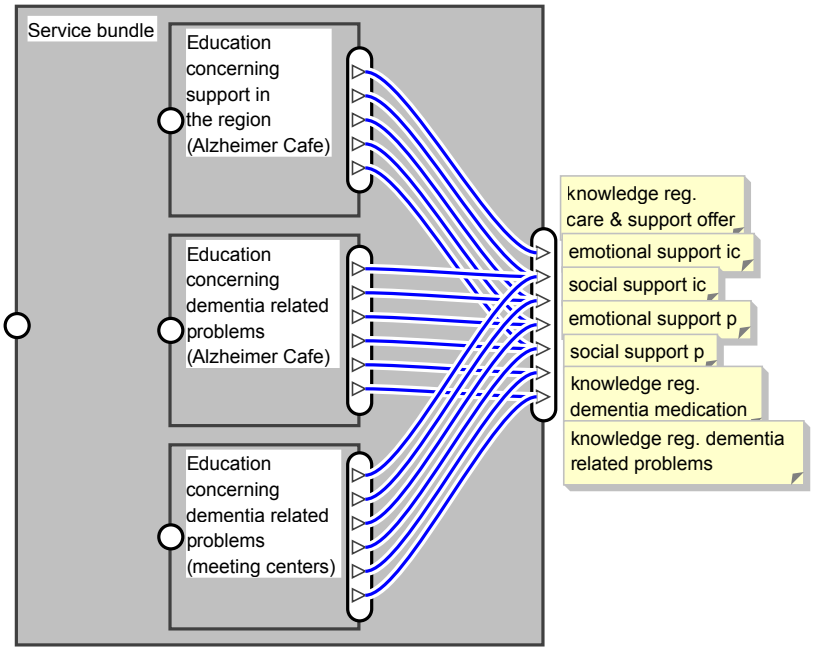

Figure 8.6: A more elaborated free-of-charge service bundle for informal carers who ask for companions contact

The role of context information in *Serviguration.*
Also the role of context information is demonstrated in this example scenario. One of the four services that can satisfy the given demand is "discussion group for informal carers", provided by the GGZ. Besides the GGZ, also meeting centers provide the same service, resulting in the same service outcomes (but unlike the service of the GGZ, no fee is required). Theoretically, this service of meeting centers would also be a suitable solution. As explained before, the main idea behind meeting centers is combination therapy, for people with dementia and for their informal carers (with the exception of the service "education concerning dementia related problems" which is offered with no such limitation). The input for the *serviguration* process included a demand and context information. The latter specified that only an informal carer participates in this scenario, and no patient. As the discussion group service of meeting centers is provided only as combination therapy, it was disqualified as a solution for the current scenario, although it provides the desired resources.

8.4 Study Analysis

Studies in ontology theory have a dual goal: validating an ontology and solving a problem of the domain at hand.

8.4.1 Analysis from an Ontology Development & Validation Perspective

Ontology validation
Our claim is that having modeled supply and demand as described in the previous section, using our service ontology, allows us to automate the process of creating service bundles for a given set of customer demands.

In order to validate our claim we need to generate service bundles with an algorithm that uses the underlying service ontology as a basis, and answer two questions: (1) whether the generated service bundles offer a good solution for the demands at hand, and (2) whether all suitable solutions (service bundles) have been generated. This validation process should take place twice, such that the above questions are answered by domain experts as well as by actual customers. The latter requires an operational prototype information system, which is currently not available within the FrUX project. Therefore our validation with domain experts imitated the *serviguration* algorithm by means of pen and paper and large Excel sheets. Given a set of services that have been modeled, we defined a number of test cases, each including one or more customer demands. Using demands as triggers for the *serviguration* process, for every test case we generated all the service bundles that can satisfy the given demand(s). Due to the causal nature of production rules, we could easily explain why every service is or is not included in a service bundle.

Next, we presented the generated service bundles to domain experts and asked them to answer the two questions described above. In every case where one of the questions was answered negatively, we analyzed why this happened, and in all these cases we found that the shortcoming is due to wrong modeling decisions in the first place: either inaccurate production rules (considering demands/resources while neglecting to consider their properties), or wrong service dependencies were modeled by domain experts. We corrected these and generated new service bundles in a second and a third iteration of the validation process. This time domain experts declared that in all test cases all the generated service bundles offer a good solution for the given customer demands, and all the desired service bundles were generated. Hence our claim proved to be correct in this study.

The test cases we performed in our validation process involved a variety of demands, resources and services. Yet, the underlying principle was shared: *Serviguration.* We presented an interesting test case (involving an often heard demand)

elaborately in the previous section. The test case resulted in the service bundles in Figures 8.4, 8.5 and 8.6.

Lessons on the *serviguration* ontology
We also gained important insights regarding the development of the ontology from this study.

First, as explained earlier in this chapter, we observed that the *functionality* of a service is not good enough a criterion for searching and selecting services to meet customer demands, because multiple services with the same functionality can deliver different benefits. This supports our approach, where the benefits of a service are criteria for selecting services to include in a service bundle.

Second, this study played an important role in understanding the role of *context* information in the matchmaking between customers and available services. Due to the nature of the study domain, where every customer is different (unlike, for example, in the energy domain where characteristics of market segments are determinants for selecting services to offer to customers), context information is quite complex and divergent in this study. Since a supplier perspective on service offerings is still more common, it often seems natural to say that the selection of *a service* depends on the context of the customer. Yet, this study helped us understand how sometimes not a service as a whole is context dependent, but in fact the selection of a benefit to search – and not a service – is context dependent. Benefits, in turn, determine the service we offer to the customer, because different benefits are offered by different services.

Third, this study served us for an analysis of conflict management in production rules between demands and resources (as explained in Section 5.3.1). Ideally, conflict management should be automated. Yet we found that while certain conflicts between production rules can be classified as "non-solvable" (hence there is no solution), we could not find any pattern in how domain experts solve other conflicts. Consequently, their resolution cannot be performed by software, and a human intervention is required. It may be possible that conflict resolution could be performed by software if we define domain specific production rules. Yet, an ontology like ours is intended to be domain-independent, and therefore we did not investigate this.

Lessons on modeling complex domains
Domain complexity is a main modeling obstacle in realizing software support for real-world situations. As knowledge of these domains is possessed by domain experts, ideally they should model it. However, they are often not accustomed to structured modeling techniques. Therefore we modeling experts were engaged in a cooperative and iterative modeling process with domain experts, and often assumed the role of a helpdesk for modeling decisions. As our validation process has shown, mistakes are hard to avoid completely when modeling substantial amounts of information. Modeling mistakes are the result of (1) domain complexity, (2) the difficulty in making implicit domain knowledge explicit (business logic is often implicit, and

exists only in the minds of humans) and (3) domain experts' lack of experience in structured modeling techniques.

We learned that two solutions can tackle these problems. First, an important lesson from this study and other studies is that domain experts require a learning process in which they model their domain *together* with knowledge modeling experts. Second, software tools can help domain experts, for example by providing alerts when domain experts make wrong or suspicious modeling decisions (such alerts are supported by the constraints we list and formalize in Appendix A). The current study became quite cumbersome at times due to the absence of software support. In earlier studies, where we used the OBELIX tool support (discussed in Chapter 6), domain experts reacted enthusiastically.

Another lesson concerns our ability to reduce domain complexity. One of the problems the health and welfare sector is facing, and for which our service modeling approach should offer a solution, is the large number and variety of available health care and welfare services, available for clients: patients and their carers. With such a large number of available services, finding suitable services to satisfy a client's demand becomes a difficult task, especially for elderly people. The number of services is high, and complex restrictions exist for the consumption of services as a bundle. Our service ontology is used in this study to reduce this complexity for clients. In configuration terms, the *possible configuration space* is enormous, and the task of finding *suitable configurations* is burdensome. In our validation process described above we defined several test cases, each with different customer demands, and generated service bundles that satisfy these demands. In all but one test case, the number of generated bundles per test case was lower than ten. One test case yielded 29 service bundles, of which the majority included the same three services, plus one or more of five repeating services that can be added to the core three services. In sum, using a dataset of 38 services, where the number of possible service bundles is as high as 274,877,906,943 ($2^{38} - 1$), our service ontology succeeded in reducing the task complexity to ten solutions only.

8.4.2 Domain Experts' Reflection on the Study

The service ontology facilitates focusing on customers
Domain experts modeled two perspectives on services: (1) needs, wants and demands of persons with dementia and their carers (customer perspective), and (2) the actual service offerings (supplier perspective). This modeling effort was different from their common practices in two ways. First, it required investigating the dementia care field in great detail, much more elaborately than they were used to. Second, they often used to advise on help from a service offering perspective (such that knowledge on the available services is the starting point) instead of what a client actually asks

for. The latter was the starting point for offering services to customers in the study presented here.

Recently, the health care sector has been changing from using a supplier perspective to using a customer perspective. For example the formal indication system (to determine the entitlement of patients for health care services) has been changed from a supply perspective system to a system that takes the needs of a client as a starting point. To be able to understand which services provide a good response to demands of clients, it was necessary to look more to what specific outcomes different services provide. This is a rather new way in this field. The service ontology made this possible because it focuses on selecting services (as solutions for a customer demand) based on their outcomes.

The service ontology as a means to learn a domain in detail
The recently developed National Dementia Program (NDP) (Meerveld et al. 2004) was used by domain experts as a starting point to describe needs, wants and demands of people with dementia and their carers. Fourteen problem areas are defined in the NDP, and possible solutions are presented for each of these problems. However, these possible solutions do not always match with the problem area. For example, in problem area "what is the problem and what can help?" one would expect solutions in terms of information, diagnostics and possible care and welfare solutions. Besides these solutions, the NDP also offers solutions by treatment itself. However, this is a solution for other problem areas. Furthermore, when the solution "advice" is presented in the NDP, the information regarding this advice is not consistent, sometimes information is presented on by whom the advice is given, other times information on the type of advice.

Therefore, to be able to make a proper match between demands and service offerings, more precise and detailed information is needed. A lot of detailed information had to be gathered to model needs and service offerings, as we describe in Section 8.3. During this process shortcomings in the NDP have been identified, and detailed information on service offerings has been gathered. Thus domain experts gained new insights into their own domain, thanks to using our structured modeling approach. This structured approach forces domain experts to make explicit the implicit rules and regulations behind the offering of every service. Consequently, it forces domain experts to ask questions that they would not ask otherwise. This results in a better understanding of the available services, so that domain experts improve their ability to define a suitable offering for customers.

DEM-DISC's target group requires detailed information on service offerings
In spite of its shortcomings, the NDP serves as a guideline for practitioners. Yet, as our study showed, understanding the intricacies of various service offerings (supplier perspective) is very complex even for domain experts. First, in accordance with the shift to customer-oriented health care, this study concentrates on the *outcomes* of

services. Detailed information was needed to build the model of available services.

However, the information on which service is offered on what conditions was not easily available. General information on services is provided via guides, brochures and the Internet. For more detailed information domain experts contacted the agencies providing the services. Formal criteria set by the agencies were not always used in practice. For example, a service to help persons with dementia with their financial administration at home is formally not available for persons who are rich enough to be able to hire a person from a private company. However, the welfare professional may be willing to provide this service if he thinks that this gives him an opportunity to keep a finger on the pulse in the situation of the person with dementia at home. Second, in other cases the agencies themselves were unable to provide precise information on criteria for use of the service, and domain experts were sent from pillar to post. This is alarming if one realizes that domain experts are familiar with the care and welfare sector, while patients and carers who need this information most often are not.

Therefore, domain experts involved in this study are only more convinced that a dynamic interactive social chart would be a valuable contribution for persons with dementia and their (in)formal carers. DEM-DISC, for which they have modeled a knowledge base, is intended to be such a dynamic interactive social chart, based on the service ontology presented in this thesis.

Developing software support
Human-computer interaction is am important factor in the design of information systems, especially when the target group is not accustomed to using information systems (as in our case). While our study did not investigate human-computer interaction issues, we gained some insights that can be of assistance in designing interfaces. We found that in many cases when we design service bundles for a given set of customer demands, there is a core of services that appear in most service bundles that satisfy the given customer demands, in addition to a number of additional services that may be added. Solution service bundles can be presented to customers similarly, making a distinction between a basic bundle (including those services that repeatedly appear in solution bundles) and additional services (those that distinguish between bundles).

8.5 Study Discussion from a Broader Perspective

We showed that automating *serviguration* is feasible; in the health sector it is also highly required
Several characteristics of the health sector make it very suitable for applying our service ontology and modeling approach:

- Customer orientation. The health sector has been shifting from a supply-side orientation to a demand-side orientation. This is in line with the policy of the Dutch government, giving customers more flexibility in managing their consumption of health care and welfare services.
- Health care and welfare services are highly intangible, often providing advice, emotional and social support. These are typically not items that can be described based on physical characteristics.
- Supply-side domain complexity. Rules for offering services are highly specific per service provider, and are also characterized by a large number of legislative regulations, which are also in constant change. A broad spectrum of highly fragmented service offerings exists.
- Demand-side domain complexity. Customers in this sector typically have complex demands for which a cross-organizational solution is required, because in the health sector service providers are highly specialized. In most cases a customer requires more than just one service of one service provider.

All these factors make this domain a very good candidate for validating our ontology. Due to domain complexity, software-aided support for offering service bundles to customers is highly desired in this domain, as domain experts have repeatedly stated throughout our study. While customers cannot see the wood for the trees, our computational model of health care and welfare services manages to decrease domain complexity, and generate suitable solutions for real-world customer problems.

The *serviguration* ontology vs. OWL-S

The incorporation of business logic in our *serviguration* ontology is what makes the ontology suitable for relieving customers from the burden of coping with domain complexity. This is opposed to the OWL-S web service ontology (OWL Services Coalition 2004). First, OWL-S does not include constructs for modeling supply-side business rules as we have described in our examples in Section 8.3.4 (referred to as service dependencies in our ontology). Second, OWL-S does not deal with customer demands; any interaction with an OWL-S service requires that users use the terminology of OWL-S service profiles and process models; these use a supply-side terminology.

Service bundling itself is a service for customers

An important business decision is which party will offer this service. A major Dutch medical insurance has already shown interest in this role. The party that offers this service will have control over the business logic that is modeled in the information system. If an insurance company has uncontrolled power, it will be able to maneuver the system's business logic, such that its own services will be offered to customers, instead of competing services. Another option is that a neutral party will offer the

service, e.g., a governmental organization or a foundation that represents the system's customers.

Beyond the current research
Implementing information systems for offering customer centric services beyond the boundaries of one enterprise is broader than what we have discussed in this chapter. Other issues are also being dealt with by us and by others in the research community.

First, the reasoning we have presented is customer-centric; it centers around finding service bundles that satisfy customer needs. However, a service bundle must also be economically interesting for all the enterprises involved in it. In Chapter 7 we describe how our approach can be used also for performing a business analysis to ensure economic feasibility of service bundles.

Second, an ontology as ours provides a means for reasoning with knowledge. It does not describe *how* the knowledge is obtained, but assumes that the knowledge is available. Knowledge has to be extracted from at least two user groups: customers and suppliers. Knowledge on possible services to include in a bundle has to be modeled in advance by enterprises that wish to participate in potential joint offerings (or by an intermediating service broker, whether commercial or governmental). Next, to be able to use the reasoning capacity that the service ontology provides, software must know what are the demands of a customer, so that a solution can be searched for these demands. Hence a human-computer interaction is needed, to model services and to obtain information on customer demands, based on the need hierarchy that the service ontology provides. This interaction between a website as DEM-DISC and customers is another topic of research within the FrUX project.

Part IV

The Final Touch

Chapter 9

Conclusions and Future Research

In this chapter we take stock of the results of this thesis. We review the key issues in the thesis (see Section 9.1), and re-examine research questions in Section 9.2 in the light of the whole thesis. In Section 9.3 we provide a future outlook, discussing future research directions. In Section 9.4 we present a broader view on the realization of e-service initiatives. Finally, in Section 9.5 we present our view on software-aided reasoning with business logic.

9.1 Key Points and Conclusions

The main research question of this thesis reads:

> "How can services be modeled such that the task of designing service bundles can be automated?"

Throughout this thesis we show how software-aided service bundling can be achieved using our formalized conceptual modeling approach. Our approach uses techniques from computer science and artificial intelligence to reason about topics from business research, and is based on understanding the following key principles, and putting them into practice.

Two perspectives on software-aided service bundling: business research perspective and computer & information science perspective.
Automated (software-aided) service bundles design is the main topic of this thesis. We employ knowledge management, artificial intelligence and requirements engineering techniques to add a layer of formalism to existing knowledge on services, the result of decennia of research within business schools. Once formalized, this knowledge can be subject to reasoning by means of software.

We achieved this formalism by developing an ontology – a formalized conceptual model – that describes services as they are interpreted in the service management and service marketing literature, published by business researchers. The ontology can be used to model (describe) services and business rules for creating composite service packages, so-called service bundles.

Two perspectives on service modeling: value exchange and configuration.
Our service ontology includes constructs to model services. Two different backgrounds were taken into consideration in the service description. First, services are interpreted in our service ontology as business researchers interpret them: economic activities, deeds and performances of a mostly intangible nature. Customers and suppliers exchange economic values in economic activities, typically such that both perceive the value they receive to be greater or more substantial than the value they sacrifice. Service description should therefore represent this exchange of values (e.g., money, capabilities, experiences and goods) between customers and suppliers.

Second, in order to use the fruits of established configuration research to 'configure' services into service bundles, the service ontology must also be designed in accordance with configuration theory. Configuration is a design task, where predefined components are used as building blocks to design (configure) a larger, complex component, based on the availability of predefined connections, and associated parameters and constraints (Mittal & Frayman 1989, Löckenhoff & Messer 1994, Gruber et al. 1996). Thus, if services should be configured as components in configuration theory, service description in the service ontology should adhere also to component description in configuration theory. Our *serviguration* ontology shows that these two perspectives are not disjoint, or not even conflicting perspectives, and can very well be combined.

Two perspectives on service offering: customer and supplier.
In a business environment where power shifts from suppliers to customers, more and more often suppliers are required to provide customer-tailored products for their customers, instead of mass customized products (goods and services). In order to design such services by means of software (for example, online), it is necessary to take the customer perspective into consideration in the design of software, so that software can reason about the suitability of solutions for customers. The traditionally used supplier perspective on products is still required as well, because it enables us to compare products, to actually design products, and to describe products in such a way that suppliers can provide them.

Two representations of a service ontology: for humans and computers.
On the one hand, the ontology has a computer-interpretable representation, so that it can serve for software development. On the other hand, the ontology has a graphical representation such that it can be used by stakeholders who have no or low affinity with computer science (typically, business people have to model domain know-

ledge in terms of the ontology). These stakeholders are not helped with computer-interpretable knowledge representations.

These issues were the main principles guiding the development of our service ontology. An important reason why a service ontology is at all required is that services, due to their intangibility, cannot be described unambiguously by their physical properties. Instead, we describe them based on the exchange of economic values that they encapsulate. One may claim that also goods can be described similarly, so in fact our ontology is not a service ontology, but a product ontology. However, goods can be (and are being) described unambiguously by customers and suppliers using their physical properties, and hence the need for a different goods description does not arise.

We successfully implemented our service ontology in software tools to validate its theoretical and computational adequacy. The tools use the underlying service ontology to model real-world services and to generate service bundles based on given customer requirements. Real-world services stem from industrial studies that we engaged in as part of the development and theoretical validation of the ontology. Two large scale studies in complex domains are described in Chapters 7 and 8.

First, a study in the energy sector uses the service ontology in a business analysis where an energy supplier was considering which services to bundle with electricity supply in order to attract new customers for the core product: electricity. We provide business analysts with a four-step method to perform a business analysis for offering cross-organizational service bundles, using our service ontology (see Chapter 7).

Second, in a completely different domain, we used our service ontology to model health care and welfare services for dementia patients and their (in)formal carers. Based on this model our project partners are currently developing software that will offer dementia patients and their (in)formal carers service bundles, based on their needs and situation.

The two studies demonstrate the two different usages of our service ontology, as presented in the introduction to this thesis. Both studies show that services *can* be described in accordance with business researchers' definition of services as well as in accordance with component definition in configuration theory, so that services can be configured by software into service bundles that fit customer needs. The studies also provide evidence of the usefulness of our modeling approach.

We provide software developers with a computer-readable representation of our ontology. Software that we developed for modeling and configuring services uses this Web-based representation (in RDFS) for communication between applications. Inherent domain constraints are formalized in Appendix A, and were implemented in our software. They are an integral part of our ontology, and they must be implemented in any software application that is based on our ontology.

In fact, we show that intangibles can be configured using the same algorithms as tangibles, once they are described based on other characteristics than their physical properties. Since these intangibles are business activities, we describe them by the economic value exchange that they encapsulate. Accordingly, our contribution to business research is in providing a formal model for an existing theory that so far was expressed in natural language only, and in making this theory computational, suitable for automated support.

9.2 Reviewing the Detailed Research Questions

The core answer to our main research question was presented in the previous section as a set of principles that guide our service modeling approach:

- Two perspectives on software-aided service bundling: business research perspective and computer & information science perspective.
- Two perspectives on service modeling: value exchange and configuration.
- Two perspectives on service offering: customer and supplier.
- Two representations of a service ontology: for humans and computers.

In this section we re-examine our main research question in more detail by recapitulating the discussions on our *detailed* research questions.

What are services?
A literature review showed that the term *service* has differing interpretations and definitions within and across domains. In this thesis we interpret this term as done by business researchers. Services are economic activities, deeds and performances of a mostly intangible nature. As economic activities, services are acts of exchange of economic values between customers and suppliers.

Owing to our definition of the term 'service', our service ontology describes services as exchanges of economic values, referred to as 'resources': some resources are required by the service provider, in return for making other resources available when a service is provided. It is important to distinguish between the resources that we refer to (reflecting a value exchange) and resources required for carrying out a (business) process. The former describe the added value of services for customers and suppliers (not necessarily indicating in which order actors provide value to each other), and the latter describe *how* tasks transform objects (e.g., information, physical devices) into other objects (such that inputs are required beforehand to produce outputs). To emphasize this difference, we use the term 'outcome' to describe a benefit of a service, and the term 'output' to describe the result of a process.

By explicitly adopting service definition from business research, we distinguish ourselves from fellow researchers in computer science departments, where the business aspect of services is often neglected. While a large part of the service-related work performed in computer science departments focuses on executing activities (by means of software), the higher business logic behind these activities is often not dealt with. We posit that the automation of services, as in the case of e-services, requires that this business logic is taken into consideration in developing software for service realization. To this end, it is necessary to understand and acknowledge that services are activities in which suppliers deliver economic values to customers, in return for other economic values. Our work contributes to existing service research in computer science departments in providing an understanding and a conceptual model of services as economic activities, taking both the supplier side and the customer side into consideration, as opposed to traditional work that deals with the supplier side only. This thesis can serve as a layer on top of existing work, to ensure that e-service realization is driven by business logic from a supplier's perspective as well as from a customer's perspective.

What is the business logic behind consuming and providing service bundles? Two answers are required for this question: from a customer perspective and from a supplier perspective. According to Kasper et al. (1999) services and service bundles offer a "bundle of benefits" to customers. These benefits, being the value that a service encapsulates for customers, are in fact what customers seek when they consume services (Teare 1998, Lancaster 1966). Products (services and goods) are consumed to fulfill customer demands (Kotler 1988). Therefore from a customer perspective a service bundle should present a set of benefits that – as a whole – can satisfy customer demand(s).

From a supplier perspective there exists a series of strategic reasons to bundle services, including cost effectiveness considerations, strategic partnership considerations, marketing considerations (interdependencies between services; service differentiation), competitiveness considerations (creating entry barriers) and more. In our work we do not consider the strategic reasons to bundle services because we did not develop an ontology for modeling business strategies. Instead, we are interested in the different dependencies between instances of services, defining whether and how two or more services can be bundled, without considering the higher level strategic reason for these dependencies. Such dependencies may dictate that services are sold only as a bundle, or either as a bundle or as separate services. Some services may exclude each other, and other may function as substitutes. All these business rules can be expressed by IF THEN statements, which can be implemented in software. Other important business rules for defining service bundles relate to the price of a service. Very often the pricing model of a service bundle determines that the price of the bundle is lower than the sum of the prices of the individual services in the bundle. Pricing models can be expressed by mathematical formulas such that they

can also be implemented in software to determine the price of services and service bundles. In sum, from a supplier perspective the logic behind combining instances of services into service bundles can be captured by "IF THEN"-like rules, and the price of service bundles can be represented mathematically.

How can existing computer based reasoning techniques capture business logic? In our research we did not form any new theories to describe, explain or predict the behavior of service suppliers and their customers. Instead, we used existing business research as the main source of domain knowledge. An important obstacle in designing information systems using knowledge from business research is that this knowledge is mostly available in natural language only, and is therefore not suitable for machine-processing. A precondition for designing such information systems is that knowledge is formally represented using a machine-interpretable representation, such that software can reason about it. The use of conceptual models and ontologies to formalize domain knowledge has broadly been accepted in information science and computer science. In this thesis we employ these techniques to model knowledge that stems from the service marketing and service management literature. We show that the service bundling task can in fact be represented as a component configuration task, for which a large amount of research exists. We map our service ontology with a configuration ontology, and can consequently use existing configuration algorithms for service bundling.

Our work on service configuration adds up to existing work on product (goods) configuration and on web service composition where similar and other issues are dealt with. The configuration of physical goods is traditionally based on physical characteristics, and it does not take higher-level business logic into consideration, as we do. Web services are applications that can realize *services* over the Web; *services* are economic activities. Hence, the notion of economic value should be present in web services too. Yet, existing work on web service composition takes into consideration only descriptive, functional and structural features of services (Gómez-Pérez et al. 2004), and not the value-related business logic behind the actual service (economic activity) that web services realize. Our contribution to computer science and information science research is in demonstrating how business logic, although traditionally ill-defined in computational terms, can be tackled by existing computer based techniques, such that the business logic behind transactions can be supported computationally in automated (possibly online) configuration of services. We firmly believe that realizing online service offerings requires that business logic is handled on top of existing work on configuration and web services.

9.3 Future Research

The road that will lead us to a broad application of online cross-organizational service bundling is still long. We strongly believe that a multidisciplinary approach has to be adopted for the goal to be achieved. Most importantly, online service bundling requires the integration of a business perspective and an ICT perspective. Several parts of the road have been explored by us and by others, but yet there is much to explore, and the various parts have to be glued into a seamless whole.

This thesis is a pioneers' work because preceding research on offering services and service bundles has been performed in business schools, whereas we conducted research within a faculty of sciences, in a multidisciplinary environment. Unlike researchers from business schools, researchers from computer science groups use ontologies and other computer-based reasoning and knowledge representation techniques for online initiatives, often overlooking the business value perspective of such initiatives. Our work attempts to bring together these two perspectives. In this section we provide an overview of open questions for future research.

Strategic reasoning. Our service ontology includes constructs for modeling rules for bundling services, for example substitution or service enhancement. However, the ontology assumes that these rules – often based on strategic considerations – are known, and it does not model knowledge on the higher reasons for such business rules, for example product differentiation, strategic alliances or cost effectiveness. Consequently, it is impossible (using our ontology) to automate the reasoning about strategic considerations behind a service bundle. To achieve this, our ontology will have to be extended with an ontology for modeling business strategies, based on which the business rules that we use are derived.

Negotiations. Pricing models as we describe in the service ontology are suitable for cases in which a supplier determines the pricing model of a service (or a variety of pricing models, out of which customers can choose one). This is the traditional way of doing business. Nowadays power is shifting from suppliers to customers, and often the price is a result of a negotiation process. Such negotiations are beyond the scope of the current ontology.

Understanding customer behavior. The ontology presented in this thesis is generic so that it can be applied across domains. We use the economic principle that a customer is interested in the value/benefits that a service provides, rather than in the service itself. In spite of the general applicability of this principle, marketing researchers have been publishing a wealth of research on factors that influence customer behavior. The means-end theory (Gutman 1982, Zeithaml 1988) is a broadly accepted marketing theory for explaining why customers seek specific good/service attributes and benefits, by linking these attributes and benefits to customer values, defined as "consequences for which a person has no further (higher) reason for preference"

(Gutman 1982). A means-end chain is a model that "seeks to explain how a product or service selection facilitates the achievement of desired states" (Gutman 1982): customers seek means to achieve their ends (goals). Similarly to the service value (customer) perspective of our service ontology, also the means-end theory uses a goal oriented hierarchical model to understand customer behavior. Embedding the means-end theory or similar frameworks in our work therefore appears to be a promising research direction for relating solutions (service bundles) to customer needs in a more flexible way than the current service ontology allows. Our approach adds two main features that are not available in the means-end theory:

1. The means-end theory does not consider the possible *solutions* for customer needs. Customer needs are refined to the degree of desired product attributes, but these are not linked further to any elements that provide these attributes. The service ontology, on the other hand, includes both customer needs and available solutions. By using production rules as in our ontology it becomes possible to relate not only product attributes, but also possible solutions (i.e., available service offerings) to a customer's needs and values.

2. Our approach adds mechanisms for software-aided reasoning, which are not present in the means-end theory. By using AND/(EX)OR refinements in hierarchies we enable a much more detailed and useful analysis of relations than a means-end model allows. Such knowledge cannot be inferred from means-end hierarchies in their traditional form.

Also, service quality has been a very fruitful area of research for many years. At least two widely accepted generic models for defining service quality are used in business science: that of the Nordic school (Grönroos 2000) and that of the North American school (SERVQUAL, see (Zeithaml et al. 1990)). Other researchers investigated service satisfaction (which is influenced by the perceived service quality) (Matzler 2002, Liljander & Strandvik 1997). With the rise of e-services, in recent years researchers have been investigating also e-service quality, compared to traditional service quality research (Parasuraman et al. 2005, van Riel et al. 2001). Embedding this research in our service ontology may enable a much richer and precise understanding of customer behavior as a means for a coarse definition of customer requirements as input for the actual service bundling.

Process configuration and web service configuration. Software that is based on our ontology can design service bundles, artifacts that are an exchange of values between customers and suppliers, describing only the *what*, and not the *how*. Business process models describe *how* such artifacts are made operational. In the context of online scenarios it is likely that large parts of the business process will be executed by Internet-based information systems, for example using web services. Just as a

variety of service bundles can satisfy the same customer need, sometimes also a variety of (composite) business processes can operationalize the same service bundle, and a variety of (composite) software components (e.g., web services) can execute the same business process. In accordance with our discussion in Section 3.2.2, we view service configuration, business process configuration and web service configuration/composition as three separate topics of research. Relating the three to each other is a next step in bringing together the business perspective and the ICT perspective on online service bundling. More specifically, an interesting future research direction is to investigate how work on web service configuration, e.g., Omelayenko (2005), can be related to this thesis, so that once an online service bundle has been designed, its execution can follow over the Web. We explore some ideas in this direction in Pedrinaci, Baida, Akkermans, Bernaras, Gordijn & Smithers (2005).

Performance. Domain complexity, rather than computational complexity, has been the main consideration in our work. First and foremost, the ontology had to enable a software-aided process where answers can be given for all relevant (domain) questions that users may pose. Configuration tasks may have to cope with performance problems, caused by a large number of components in the problem-solution models and by the large possible ways to configure these components. Configuration performance, caused by computational complexity, is a research topic in its own right. Especially in online scenarios customers are not patient, and a quick solution is required (but then again: what is 'quick'?).

9.4 E-Services: a Broader View

Authors have been writing about e-services in the last few years from different perspectives. Many of them focus on the technical aspect of e-services (e.g., Kreger (2003), Andrews et al. (2003), Edmond & ter Hofstede (2000) and McIlraith et al. (2001)), while others use a business value viewpoint to examine e-services (e.g., Xue et al. (2003) and Bolton (2003)) or services in general (whether Web-based or not, e.g., Barrutia Legarreta & Echebarria Miguel (2004), Lovelock (1983) and Normann (2001)).

Although the service ontology presented in this thesis is generic such that it applies also to traditional real-world (non Web-based) services, its importance for e-services is greater, as in e-services all knowledge must be formalized to enable reasoning by means of software. This is what makes e-services a truly multidisciplinary field. Businesses offer and consume e-services to make money, to achieve their business goals and to realize their business strategies. Eventually, this is partly realized by software components. A major challenge lies in ensuring that these software components are indeed a true reflection of business goals and business strategies. To this

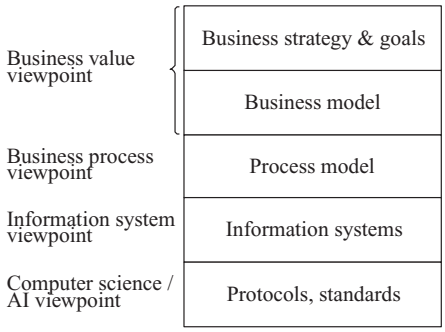

Figure 9.1: An architectural approach to e-services

end, the various aspects of e-services must be intertwined. This thesis is another step in the road that leads to multidisciplinary e-service realization.

Rust & Kannan (2003) argue that the e-service paradigm provides "an approach that helps develop strong business strategies and build customer equity" by focusing on customer satisfaction, customer-tailored products and 1-to-1 marketing as a means for expanding revenues. This is opposed to traditional e-commerce, referred to as *e-tailing*, which focuses on efficiency and on selling commodities as a means to reduce costs. E-Services are thus seen as enablers of new business strategies. Yet, are these expectations different from the expectations we had from e-tailing, until the dotcom bubble burst, and Nasdaq experienced a free fall?

Therefore we firmly believe that a thorough analysis and implementation of e-service activities is required for the successful realization of e-service initiatives; an analysis and an implementation in which every viewpoint is reflected in related viewpoints. Business strategies and business goals are reflected in business models. Business models on their turn are transformed into business process models, which are partly technical process models, executed by information systems that rely on technical, human and information infrastructures. While each of these fields is a respectable research field in its own right, we believe that the key to successful implementation and utilization of e-services is in relating these fields, such that principles, guidelines and decisions from one viewpoint are reflected also in other viewpoints. In an architectural approach where the business value viewpoint is the top layer, followed by the business process viewpoint, the information system viewpoint and finally the computer science/AI viewpoint, the contribution of this thesis is in making part of the business value viewpoint accessible for the underlying viewpoints. As a result, principles, guidelines and decisions from the business value viewpoint can be reflected also in underlying viewpoints (see Figure 9.1), such that every viewpoint realizes requirements and principles from higher viewpoints.

9.5 Our View on Software-aided Reasoning with Business Logic

Business logic encompasses a series of rules concerning a domain. In software development business logic often refers to rules to manipulate the data stored in an information system. Rules of operational nature are typically well-defined in company procedures, and require complete knowledge (of the matter at hand). Examples are rules for customer registration, for order shipment and for payment transactions.

High-level (not of operational nature) business logic, however, is much less well-defined, and is more often available only implicitly, in the minds of humans. Scozzi & Garavelli (2005) argue that business development, design and analysis – activities that require reasoning with high-level business logic – are "highly unstructured and characterized by difficult-to-forecast activities linked by reciprocal rather then sequential dependencies". This explains the low number of research efforts to develop software-based reasoning methods involving high-level business logic, and the need for human intervention in such software-aided scenarios.

Our thesis explicitly deals with high-level business logic, because high-level (supply-side and demand-side) business logic is the only logic that can explain the provisioning and consumption of services. Nevertheless, also we limit ourselves, and did not model the background of business rules, such that our ontology does not support software-based strategic reasoning.

As we concluded in Chapter 7, software support has its limitations, because certain decisions cannot be automated. Not yet, at least. But then again, who thought in the 1980's that automation will go as far as it has gone today?

We therefore see many research challenges and possibilities in stretching the boundaries of software-aided 'soft' tasks as decision making on the level of business strategy and business development.

From a business research perspective, this line of research requires developing and formalizing conceptual models of the management process, decision making, business design & development and business strategies. From a computer science perspective, good starting points are requirements engineering (RE) and artificial intelligence (AI) research on managing uncertainty and vagueness. Requirements engineers have gained valuable experience in eliciting knowledge from system stakeholders, where this knowledge often shares important characteristics with high-level business logic: it is ill-defined and available in the minds of humans. Therefore we propose to use RE techniques for the development of conceptual models for high-level business logic. Similarly, the AI community has gained experience in reasoning with vagueness and uncertainty. This research may prove valuable for reasoning with high-level business logic, as business logic is often vague (try to concretize and oper-

ationalize the term ‘good business strategy’) and subject to uncertainty (for example, a company’s performance depends also on its competitors performance).

Appendix A

Inherent Constraints in the Service Ontology

A.1 Introduction

Some knowledge does not appear explicitly in the UML diagrams of our service ontology and in the RDFS implementation thereof, because it is considered to be inherent to the domain, and because it cannot be expressed using these representation techniques. Yet, making this knowledge explicit becomes important when information systems must reason about a domain. Therefore, in this appendix we list constraints that are inherent to the service ontology, and express them using a computer-processable notation, to facilitate automation. All constraints are described using first-order predicate logic. These constraints are an integral part of our service ontology.

We use the following terminology in the rest of this appendix:

- The term "input port" should be interpreted as "a service port in an input interface".
- The term "outcome port" should be interpreted as "a service port in an outcome interface".
- The expression "input port of service x" should be interpreted as "a port in the input interface of service x".
- The expression "outcome port of service x" should be interpreted as "a port in the outcome interface of service x".

- The relation "a service bundle includes one or more service elements" is not transitive. For example, any of the service elements $\{SE_1, SE_2, \ldots SE_n\}$ included in service bundle SB_x may be a service bundle itself. Imagine that SE_2 is a service bundle, and it includes service element SE_{n+1} (and possibly other service elements). We then say that "SB_x includes SE_2" and that "SE_2 includes SE_{n+1}", but this does not imply that "SB_x includes SE_{n+1}".
- The expression "service port x is linked with service port y" should be interpreted as "a service link connects service port x with service port y", not implying whether the service link starts at x and ends at y or vice versa.
- A "service model" is a valid instantiation of the service ontology, including concepts and relations between these concepts, as defined in the service ontology.

A.1.1 Universe of Discourse

In the rest of our discussion, our Universe of Discourse is a (any) service model SM. We define the following sets:

- SE is the set of all service elements in SM (note: as defined by the service ontology, the concept 'service element' has two subtypes: 'elementary service element' and 'service bundle').
- SP is the set of all service ports in SM.
- SL is the set of all service links in SM.
- SB is the set of all service bundles in SM.
- R is the set of all resources in SM.
- Q is the set of all service properties in SM.
- PM is the set of all pricing models in SM.
- D is the set of all demands in SM.

A.1.2 Predicates

We define the following predicates:

- CONNECTS_PORTS is a ternary predicate between a service link and two service ports, indicating that the specified service link connects the two specified service ports.

- STARTS_AT_PORT is a binary predicate between a service link and a service port, indicating that the specified service link starts at the specified service port.

- ENDS_AT_PORT is a binary predicate between a service link and a service port, indicating that the specified service link ends at the specified service port.

- PORT_OF_SERVICE is a binary predicate between a service port and a service element, indicating that the specified service port belongs to a service interface that belongs to the specified service element.

- DIRECT_COMPONENT_OF is a binary predicate between a service element and a service bundle, indicating the inverse relation to "service bundle includes service element" that we discussed earlier in this appendix.

- IS_INPUT is a unary predicate indicating that the specified service port belongs to an input interface (without specifying the service element to which the service port and service interface belong).

- IS_OUTCOME is a unary predicate indicating that the specified service port belongs to an outcome interface (without specifying the service element to which the service port and service interface belong).

- DELIVERS_INPUT_FOR is a binary predicate between two service elements A and Z indicating that service A provides a service outcome which is used as an input by some service B, and service B provides a service outcome which is used as an input by some service C, and... some service Y provides a service outcome which is used as an input by service Z". More formally, either (1) a service link exists that starts at A and ends at the Z, or (2) there exist a third service element B such that a service link exists that starts at A and ends at the B, and also DELIVERS_INPUT_FOR(B, Z).

- REQUIRES_RESOURCE is a binary predicate between a service port and a resource, indicating that the specified resource is assigned to the specified service port.

- GREATER_OR_EQUAL is a binary predicate between two resources, indicating that the first resource is greater than or equal to the second resource.

- EQUAL_RESOURCES is a binary predicate between two resources, indicating that the first resource is equal to the second resource.

- GREATER_THAN_RESOURCE is a binary predicate between two resources, indicating that the first resource is greater than the second resource.

- INCLUDES_LINK is a binary predicate between a service bundle and a service link, indicating that the specified service link connects service ports included in the specified service bundle (note: this leaves two options open: either the service link connects two service ports of two distinct service elements included in the service bundle, or it connects a service port of the service bundle with a service port of a service element within the bundle).

- CE is a binary predicate between two sets of one or more service elements, indicating that the *core/enhancing* service dependency exists between these two sets, such that the service dependency starts at the first set, and ends at the second set.

- CS is a binary predicate between two sets of one or more service elements, indicating that the *core/supporting* service dependency exists between these two sets, such that the service dependency starts at the first set, and ends at the second set.

- OB is a binary predicate between two sets of one or more service elements, indicating that the *optional bundle* service dependency exists between these two sets, such that the service dependency starts at the first set, and ends at the second set.

- BU is a binary predicate between two sets of one or more service elements, indicating that the *bundled* service dependency exists between these two sets, such that the service dependency starts at the first set, and ends at the second set.

- EX is a binary predicate between two sets of one or more service elements, indicating that the *excluding* service dependency exists between these two sets, such that the service dependency starts at the first set, and ends at the second set.

- SU is a binary predicate between two sets of one or more service elements, indicating that the *substitute* service dependency exists between these two sets, such that the service dependency starts at the first set, and ends at the second set.

- PM_AT_PORT is a binary predicate between a pricing model and a service port, indicating that the specified pricing model is assigned to the specified service port.

- PM_AT_LINK is a binary predicate between a pricing model and a service link, indicating that the specified pricing model is assigned to the specified service link.

- IS_MONETARY is a unary predicate indicating that the specified resource is of type 'monetary resource'.
- EQUAL_RES_NAMES is a binary predicate between two resources, indicating that both resources have the same name.
- EQUAL_RES_TYPE is a binary predicate between two resources, indicating that both resources have the same type.
- DESCRIBES_RESOURCE is a binary predicate between a service property and a resource, indicating that the specified resource is described by the specified service property.
- IS_COMPARABLE is a unary predicate indicating that the specified service property is comparable.
- EQUAL_PROP_NAMES is a binary predicate between two service properties, indicating that both service properties have the same name.
- EQUAL_PROP_VALUE is a binary predicate between two service properties, indicating that both service properties have the same value.
- GREATER_NUM_VALUE is a binary predicate between two service properties, indicating that both service properties have a numeric value, and that the value of the first service property is greater than the value of the second service property.
- EQUAL_PROP_UNITS is a binary predicate between two service properties, indicating that both service properties use the same unit to describe their value.
- EQUAL_PROP_TYPE is a binary predicate between two service properties, indicating that both service properties use the same datatype for their units.
- IS_NUMERIC is a unary predicate indicating that the specified service property has a numeric value.
- SELECTION is a quaternary predicate with four arguments: (1) a demand; (2) a (possibly empty) set of service properties that describe this demand; (3) a resource; and (4) a (possibly empty) set of service properties that describe this resource. It indicates that there exists a *selection* production rule between the demand, described by its service properties, and the resources, described by its service properties.
- REJECTION is a quaternary predicate with four arguments: (1) a demand; (2) a (possibly empty) set of service properties that describe this demand; (3) a resource; and (4) a (possibly empty) set of service properties that describe this

resource. It indicates that there exists a *rejection* production rule between the demand, described by its service properties, and the resources, described by its service properties.

- POSITIVE is a quaternary predicate with four arguments: (1) a demand; (2) a (possibly empty) set of service properties that describe this demand; (3) a resource; and (4) a (possibly empty) set of service properties that describe this resource. It indicates that there exists a *positively influenced by* production rule between the demand, described by its service properties, and the resources, described by its service properties.

- NEGATIVE is a quaternary predicate with four arguments: (1) a demand; (2) a (possibly empty) set of service properties that describe this demand; (3) a resource; and (4) a (possibly empty) set of service properties that describe this resource. It indicates that there exists a *negatively influenced by* production rule between the demand, described by its service properties, and the resources, described by its service properties.

A.2 Constraints Related to Service Links

A service link is a connection between two service ports, such that it starts at one service port and ends at another.

- A service link may not exist between two service ports that belong to the same service element (note: this also implies that a service port cannot be linked to itself).

$$\forall x{\in}SL.\forall y{\in}SP.\forall z{\in}SP.\ [(STARTS\_AT\_PORT(x, y) \wedge ENDS\_AT\_PORT(x, z)) \rightarrow \neg\exists i{\in}SE.(PORT\_OF\_SERVICE(y, i) \wedge PORT\_OF\_SERVICE(z, i))]$$

- A service link may exist between two service ports y and z belonging to service elements i and j respectively only if (1) i and j are included in the same service bundle, or (2) i includes j, or (3) j includes i.

$$\forall x{\in}SL.\forall y{\in}SP.\forall z{\in}SP.\forall i{\in}SE.\forall j{\in}SE.\ [(CONNECTS\_PORTS(x, y, z) \wedge PORT\_OF\_SERVICE(y, i) \wedge PORT\_OF\_SERVICE(z, j)) \rightarrow \exists q{\in}SB.(DIRECT\_COMPONENT\_OF(i, q) \wedge DIRECT\_COMPONENT\_OF(j, q)) \vee DIRECT\_COMPONENT\_OF(j, i) \vee DIRECT\_COMPONENT\_OF(i, j)]$$

- No cycles between services are allowed. A cycle is a situation involving two or more services such that service A provides an input for service B; service B provides an input for service C;... provides an input for service N; service N provides an input for service A.

  $$\forall i \in SE. \forall j \in SE. [DELIVERS\_INPUT\_FOR(i, j) \rightarrow \neg\, DELIVERS\_INPUT\_FOR(j, i)]$$

  whereby:

  $$\forall i \in SE. \forall j \in SE.$$
  $$[DELIVERS\_INPUT\_FOR(i, j) \leftrightarrow \exists x \in SL. \exists y \in SP. \exists z \in SP.$$
  $$((STARTS\_AT\_PORT(x, y) \wedge ENDS\_AT\_PORT(x, z) \wedge$$
  $$PORT\_OF\_SERVICE(y, i) \wedge PORT\_OF\_SERVICE(z, j)) \vee$$
  $$\exists a \in SE. \exists b \in SP. (STARTS\_AT\_PORT(x, y) \wedge ENDS\_AT\_PORT(x, b) \wedge$$
  $$PORT\_OF\_SERVICE(y, i) \wedge PORT\_OF\_SERVICE(b, a) \wedge$$
  $$DELIVERS\_INPUT\_FOR(a, j)))]$$

Service links may connect service ports of two service elements differently, based on the service interface to which the service ports belong and based on the question whether one of the service elements includes the other. We distinguish between the following cases:

- An input port of service i may be linked to an outcome port of service j only if i and j are included in the same service bundle. The service link starts at the outcome port, and ends at the input port.

  $$\forall x \in SL. \forall y \in SP. \forall z \in SP. \forall i \in SE. \forall j \in SE.$$
  $$[(PORT\_OF\_SERVICE(y, i) \wedge IS\_INPUT(y) \wedge$$
  $$PORT\_OF\_SERVICE(z, j) \wedge IS\_OUTCOME(z) \wedge CONNECTS\_PORTS(x, y, z)) \rightarrow$$
  $$(STARTS\_AT\_PORT(x, z) \wedge ENDS\_AT\_PORT(x, y) \wedge$$
  $$\exists q \in SB. (DIRECT\_COMPONENT\_OF(i, q) \wedge DIRECT\_COMPONENT\_OF(j, q)))]$$

- An input port of service i may be linked to an input port of service j only if i includes j or j includes i. In both cases the service link that connects both ports starts at the service that includes the other.

  $$\forall x \in SL. \forall y \in SP. \forall z \in SP. \forall i \in SE. \forall j \in SE.$$
  $$[(PORT\_OF\_SERVICE(y, i) \wedge IS\_INPUT(y) \wedge$$
  $$PORT\_OF\_SERVICE(z, j) \wedge IS\_INPUT(z) \wedge CONNECTS\_PORTS(x, y, z)) \rightarrow$$
  $$(DIRECT\_COMPONENT\_OF(j, i) \wedge STARTS\_AT\_PORT(x, y) \wedge$$
  $$ENDS\_AT\_PORT(x, z)) \vee (DIRECT\_COMPONENT\_OF(i, j) \wedge$$
  $$STARTS\_AT\_PORT(x, z) \wedge ENDS\_AT\_PORT(x, y))]$$

- An outcome port of service i may be linked to an outcome port of service j only if i includes j or j includes i. In both cases the service link that connects both ports ends at the service that includes the other.

  $\forall x \in SL. \forall y \in SP. \forall z \in SP. \forall i \in SE. \forall j \in SE.$
  $[(PORT\_OF\_SERVICE(y, i) \wedge IS\_OUTCOME(y) \wedge$
  $PORT\_OF\_SERVICE(z, j) \wedge IS\_OUTCOME(z) \wedge CONNECTS\_PORTS(x, y, z)) \rightarrow$
  $(DIRECT\_COMPONENT\_OF(j, i) \wedge STARTS\_AT\_PORT(x, z) \wedge$
  $ENDS\_AT\_PORT(x, y)) \vee (DIRECT\_COMPONENT\_OF(i, j) \wedge$
  $STARTS\_AT\_PORT(x, y) \wedge ENDS\_AT\_PORT(x, z))]$

A.3 Constraints Related to Resources

- Consider two service ports y and z such that resource r is assigned to service port y, and resource s is assigned to service port z. These service ports y and z may be connected by a service link that starts at y and ends at z based on the following conditions (see definition of $r \geq s$ in Section A.7):
  - when y and z are input ports: only if $r \geq s$
  - when y and z are outcome ports: only if $s \geq r$
  - when y is an outcome port, and z is an input port: only if $r \geq s$
  - when y is an input port, and z is an outcome port: no service link as specified here (starts at y, ends at z) is allowed

  $\forall x \in SL. \forall y \in SP. \forall z \in SP. \forall r \in R. \forall s \in R.$
  $[(REQUIRES\_RESOURCE(y, r) \wedge REQUIRES\_RESOURCE(z, s) \wedge$
  $STARTS\_AT\_PORT(x, y) \wedge ENDS\_AT\_PORT(x, z)) \rightarrow$
  $(((IS\_INPUT(y) \wedge IS\_INPUT(z)) \rightarrow GREATER\_OR\_EQUAL(r, s)) \wedge$
  $((IS\_OUTCOME(y) \wedge IS\_OUTCOME(z)) \rightarrow GREATER\_OR\_EQUAL(s, r)) \wedge$
  $((IS\_OUTCOME(y) \wedge IS\_INPUT(z)) \rightarrow GREATER\_OR\_EQUAL(r, s)) \wedge$
  $\neg(IS\_INPUT(y) \wedge IS\_OUTCOME(z)))]$

A.4 Constraints Related to Service Dependencies

- The two arguments of a service dependency must be disjoint.

  $\forall i \in SE. \forall j \in SE.$
  $[(CE(i, j) \vee CS(i, j) \vee OB(i, j) \vee BU(i, j) \vee EX(i, j) \vee SU(i, j)) \rightarrow i \cap j = \emptyset]$

- Only one service dependency may exist between two service elements i and j (note: another service dependency may exist between j and i).

$$\forall i \in SE. \forall j \in SE. [(CE(i, j) \vee CS(i, j) \vee OB(i, j) \vee BU(i, j) \vee EX(i, j) \vee SU(i, j)) \rightarrow \neg \exists a \in SE. \neg \exists b \in SE. ((CE(a, b) \vee CS(a, b) \vee OB(a, b) \vee BU(a, b) \vee EX(a, b) \vee SU(a, b)) \wedge i \cap a \neq \emptyset \wedge j \cap b \neq \emptyset \wedge i \neq a \wedge j \neq b))]$$

- The notion of "X is a core service of Y" is irreflexive.

$$\forall i \in SE. \forall j \in SE. [CE(i, j) \rightarrow \neg CE(j, i)]$$

$$\forall i \in SE. \forall j \in SE. [CE(i, j) \rightarrow \neg CS(j, i)]$$

$$\forall i \in SE. \forall j \in SE. [CS(i, j) \rightarrow \neg CS(j, i)]$$

$$\forall i \in SE. \forall j \in SE. [CS(i, j) \rightarrow \neg CE(j, i)]$$

- If service i is the core service of service j, service j may not exist in a service bundle without service i. This can be generalized for the case that the arguments of service dependencies CS and CE include more than one service, e.g., CE({i1, i2, i3,...},{j1, j2, j3,...}). Then any service of the second argument may be added to any service of the first argument, and a service of the second argument may not be sold without at least one of the services in the first argument.

$$\forall i \in SE. \forall j \in SE. [(CE(i, j) \vee CS(i, j)) \rightarrow \neg b \in j. \neg q \in SB. (DIRECT\_COMPONENT\_OF(b, q) \wedge \forall a \in i. \neg DIRECT\_COMPONENT\_OF(a, q))]$$

- The service dependency *core/enhancing* between services i and j implies that j cannot be sold independently of i. Consequently, it is wrong to model the service dependency *optional bundle* in the reverse direction, as it implies that service i *may* or *may not* be sold together with service j.

$$\forall i \in SE. \forall j \in SE. [CE(i, j) \rightarrow \neg OB(j, i)]$$

- The *core/enhancing* service dependency means that a service may be combined with a core service to add value to the core service. The *excluding* service dependency means the two cannot be sold together. Hence they contradict.

$$\forall i \in SE. \forall j \in SE. [CE(i, j) \rightarrow \neg EX(j, i)]$$

- The service dependency *core/supporting* between services i and j is a strong relation, implying that j cannot be sold independently of i. Consequently, it is wrong to model the service dependency *optional bundle* in the reverse direction, as it is a weak relation, implying that service i *may* or *may not* be sold together with service j.

$$\forall i{\in}SE.\forall j{\in}SE.[CS(i, j) \rightarrow \neg OB(j, i)]$$

- The *core/supporting* service dependency means that a supporting service must be sold together with a core service. The *excluding* service dependency means the two cannot be sold together. Hence they contradict.

$$\forall i{\in}SE.\forall j{\in}SE.[CS(i, j) \rightarrow \neg EX(j, i)]$$

- The *bundled* service dependency means that one service must be sold together with another. The *excluding* service dependency means the two cannot be sold together. Hence they contradict.

$$\forall i{\in}SE.\forall j{\in}SE.[BU(i, j) \rightarrow \neg EX(j, i)]$$

- The *optional bundle* service dependency means that one service may be sold together with another. The *excluding* service dependency means the two cannot be sold together. Hence they contradict.

$$\forall i{\in}SE.\forall j{\in}SE.[OB(i, j) \rightarrow \neg EX(j, i)]$$

In Section 4.4.1 we explained why the substitution service dependency models redundant information, as far as service configuration is concerned. Still, modeling substitution is natural to many domain experts, and substitution is an important concept in business research (Porter 1985). Furthermore, the substitution dependency helps us verify that domain experts make correct use of an ontology based software tool to model business logic.

Assume we have two services i and j with a substitution service dependency SU(i, j). Based on our discussion in Section 4.4.1 substitution implies either of the following situations:

1. The service outcomes of service i are a subset of the service outcomes of service j. This is the more common interpretation of substitution.

2. If there exists a demand u which has a SEL or POS production rule with resource r of service i, then there exists also a SEL or POS production rule involving the same demand u and some resource s of service j. In other words, two services are substitutes because they provide different resources that can satisfy the same demand.

A software tool can implement this interpretation of substitution to verify whether substitution has been modeled correctly by domain experts.

$\forall i \in SE.\forall j \in SE.[SU(i, j) \rightarrow$
$(\forall r \in R.\forall y \in SP.(PORT\_OF\_SERVICE(y, i) \wedge REQUIRES\_RESOURCE(y, r) \wedge$
$IS\_OUTCOME(y)) \rightarrow \exists z \in SP.(PORT\_OF\_SERVICE(z, j) \wedge$
$REQUIRES\_RESOURCE(z, r) \wedge IS\_OUTCOME(z))) \vee$
$\exists u \in D.\exists r \in R.\exists c \in Q.\exists d \in Q.\exists y \in SP.(PORT\_OF\_SERVICE(y, i) \wedge$
$REQUIRES\_RESOURCE(y, r) \wedge IS\_OUTCOME(y) \wedge (SELECTION(u, c, r, d) \vee$
$POSITIVE(u, c, r, d)) \rightarrow \exists s \in R.\exists e \in Q.\exists z \in SP.(PORT\_OF\_SERVICE(z, j) \wedge$
$REQUIRES\_RESOURCE(z, s) \wedge IS\_OUTCOME(z) \wedge (SELECTION(u, c, s, e) \vee$
$POSITIVE(u, c, s, e))))]$

Service dependencies have two arguments; each argument is a set of one or more service elements. Two exceptions exist:

- The substitution service dependency may have only *one* service element as its *first* argument.

  $\forall i \in SE.\forall j \in SE.\forall a \in i.\forall b \in i.[SU(i, j) \rightarrow a=b]$

- The excluding service dependency may have only *one* service element in *any* of its arguments.

  $\forall i \in SE.\forall j \in SE.\forall a \in i.\forall b \in i.\forall c \in j.\forall d \in j.[EX(i, j) \rightarrow a=b \wedge c=d]$

A.5 Constraints Related to Pricing Models

- A pricing model may be assigned to a service port only if the resource assigned to this service port is of the type 'monetary resource'.

  $\forall p \in PM.\forall y \in SP.[PM\_AT\_PORT(p, y) \rightarrow$
  $\exists r \in R.(REQUIRES\_RESOURCE(y, r) \wedge IS\_MONETARY(r))]$

- A pricing model may be assigned to a service link only if the resources assigned to the service ports on both ends of the service link are of the type 'monetary resource'.

  $\forall p \in PM.\forall x \in SL.$
  $[PM\_AT\_LINK(p, x) \rightarrow \exists r \in R.\exists s \in R.\exists y \in SP.\exists z \in SP.\ (CONNECTS\_PORTS(x, y, z) \wedge$
  $REQUIRES\_RESOURCE(y, r) \wedge IS\_MONETARY(r) \wedge$
  $REQUIRES\_RESOURCE(z, s) \wedge IS\_MONETARY(s))]$

A.6 Constraints Related to Production Rules

Production rules, discussed in Chapter 5, are relations between a demand (possibly described by service properties) and a resource (also possibly described by service properties). When the same demand and resource have multiple production rules such that (1) one production rule involves the demand and resource without any service properties, and (2) one or more production rules involve the demand and resource with some service properties, the former is referred to as 'parent', and the production rules involving the demand and resource with some service properties are referred to as 'siblings'. We distinguish between four production rules: *selection*, *rejection*, *positively influenced by* and *negatively influenced by*.

- Only one production rule may exist between the same demand, resource and their sets of service properties.

  $\forall u \in D.\forall r \in R.\forall c \in Q.\forall d \in Q.\ [\text{REJECTION}(u, c, r, d) \rightarrow \neg\text{SELECTION}(u, c, r, d) \wedge \neg\text{POSITIVE}(u, c, r, d) \wedge \neg\text{NEGATIVE}(u, c, r, d)]$

  $\forall u \in D.\forall r \in R.\forall c \in Q.\forall d \in Q.\ [\text{SELECTION}(u, c, r, d) \rightarrow \neg\text{REJECTION}(u, c, r, d) \wedge \neg\text{POSITIVE}(u, c, r, d) \wedge \neg\text{NEGATIVE}(u, c, r, d)]$

  $\forall u \in D.\forall r \in R.\forall c \in Q.\forall d \in Q.\ [\text{POSITIVE}(u, c, r, d) \rightarrow \neg\text{SELECTION}(u, c, r, d) \wedge \neg\text{REJECTION}(u, c, r, d) \wedge \neg\text{NEGATIVE}(u, c, r, d)]$

  $\forall u \in D.\forall r \in R.\forall c \in Q.\forall d \in Q.\ [\text{NEGATIVE}(u, c, r, d) \rightarrow \neg\text{SELECTION}(u, c, r, d) \wedge \neg\text{POSITIVE}(u, c, r, d) \wedge \neg\text{REJECTION}(u, c, r, d)]$

- If the parents have a *rejection* production rule involving resource r, a service that provides resource r mustn't be part of a service bundle. In this case there is no logic behind modeling any other relation on the siblings level.

  $\forall u \in D.\forall r \in R.[\text{REJECTION}(u, \emptyset, r, \emptyset) \rightarrow \neg\exists c \in Q.\neg\exists d \in Q.((\text{REJECTION}(u, c, r, d) \vee \text{SELECTION}(u, c, r, d) \vee \text{POSITIVE}(u, c, r, d) \vee \text{NEGATIVE}(u, c, r, d)) \wedge c \neq \emptyset \wedge d \neq \emptyset)]$

A.7 Comparing Resources

As described in a constraint in Section A.3, we need a mechanism for comparing resources. We split the discussion on $r \geq s$ into two parts: $r = s$ and $r > s$.

According to the service ontology, a resource is described by:

- A name that describes what the resource is about (e.g., fee, energy, emotional support)
- A type (e.g., physical good, monetary resource, capability resource)
- A set of service properties. Example properties are amount, state, bandwidth, or productivity. Every service property is described by several attributes:
  - Name of the service property (e.g., amount)
  - Value of the service property (e.g., 500)
  - Unit of the service property (e.g., euro/year)
  - Value type of the service property (specifying the datatype of the value, e.g., NUMERIC/STRING)
  - Optionally a natural language description of the service property.
  - Boolean isComparable (TRUE/FALSE) indicating whether this service property plays a role in comparing the resource that it describes with other resources. When we compare two resources, only their comparable properties need to be compared.

Two resources r and s are considered to be equal if and only if they have the same name, the same type and the same comparable properties.

$$\forall r \in R. \forall s \in R. [EQUAL\_RESOURCES(r, s) \leftrightarrow$$
$$(EQUAL\_RES\_TYPE(r, s) \wedge EQUAL\_RES\_NAMES(r, s) \wedge$$
$$\forall c \in Q.((DESCRIBES\_RESOURCE(c, r) \wedge IS\_COMPARABLE(c)) \rightarrow$$
$$\exists d \in Q.(DESCRIBES\_RESOURCE(d, s) \wedge IS\_COMPARABLE(d) \wedge$$
$$EQUAL\_PROP\_NAMES(c, d) \wedge EQUAL\_PROP\_VALUE(c, d) \wedge$$
$$EQUAL\_PROP\_UNITS(c, d) \wedge EQUAL\_PROP\_TYPE(c, d))) \wedge$$
$$\forall d \in Q.((DESCRIBES\_RESOURCE(d, s) \wedge IS\_COMPARABLE(d)) \rightarrow$$
$$\exists c \in Q.(DESCRIBES\_RESOURCE(c, r) \wedge IS\_COMPARABLE(c) \wedge$$
$$EQUAL\_PROP\_NAMES(c, d) \wedge EQUAL\_PROP\_VALUE(c, d) \wedge$$
$$EQUAL\_PROP\_UNITS(c, d) \wedge EQUAL\_PROP\_TYPE(c, d))))]$$

Resource r is considered greater than resource s if and only if they have the same name and the same type, and their comparable properties are (1) equal if non-numeric, and (2) the values of comparable properties of r are bigger than those of s if they are numeric. We explicitly exclude from this definition all cases where r=s because the above definition of r>s may include cases where r=s, if all service properties of r and s are non-numeric.

$$\forall r \in R. \forall s \in R. [GREATER\_THAN\_RESOURCE(r, s) \leftrightarrow$$
$$(EQUAL\_RES\_TYPE(r, s) \wedge EQUAL\_RES\_NAMES(r, s) \wedge$$
$$\forall c \in Q.((DESCRIBES\_RESOURCE(c, r) \wedge IS\_COMPARABLE(c)) \rightarrow$$

$\exists d \in Q.(\text{DESCRIBES\_RESOURCE}(d, s) \wedge \text{IS\_COMPARABLE}(d) \wedge$
$\text{EQUAL\_PROP\_NAMES}(c, d) \wedge \text{EQUAL\_PROP\_UNITS}(c, d) \wedge$
$\text{EQUAL\_PROP\_TYPE}(c, d) \wedge$
$(\text{IS\_NUMERIC}(c) \rightarrow \text{GREATER\_NUM\_VALUE}(c, d)) \wedge$
$(\neg\text{IS\_NUMERIC}(c) \rightarrow \text{EQUAL\_PROP\_VALUE}(c, d)))) \wedge$
$\forall d \in Q.((\text{DESCRIBES\_RESOURCE}(d, s) \wedge \text{IS\_COMPARABLE}(d)) \rightarrow$
$\exists c \in Q.(\text{DESCRIBES\_RESOURCE}(c, r) \wedge \text{IS\_COMPARABLE}(c) \wedge$
$\text{EQUAL\_PROP\_NAMES}(c, d) \wedge \text{EQUAL\_PROP\_UNITS}(c, d) \wedge$
$\text{EQUAL\_PROP\_TYPE}(c, d) \wedge$
$(\text{IS\_NUMERIC}(d) \rightarrow \text{GREATER\_NUM\_VALUE}(c, d)) \wedge$
$(\neg\text{IS\_NUMERIC}(d) \rightarrow \text{EQUAL\_PROP\_VALUE}(c, d))))) \wedge$
$\neg\text{EQUAL\_RESOURCES}(r, s)]$

A.8 Concluding Remarks

In this chapter we formally describe knowledge that is inherent to the service domain, and is not modeled explicitly in UML diagrams of the service ontology and in its RDFS implementation.

Ontologies as the *serviguration* ontology are used as the basis for information systems that need to reason with knowledge modeled by the ontology. To this end, ontologies must be represented using a machine-interpretable knowledge representation. In Section 3 we presented our service ontology using UML diagrams, and we further explained and exemplified it using natural language. The ontology has also been implemented in RDFS, a Web based knowledge representation language, to facilitate automation. Yet, some knowledge is considered to be inherent to the service domain, and was not explicitly implemented in the RDFS representation of the ontology or in UML diagrams. For humans who are familiar with the domain, this knowledge is "obvious", it goes without saying (but they hardly ever make this implicit knowledge explicit). Yet, when we implement information systems, this knowledge needs to be made explicit and expressed formally, because no knowledge is "obvious" for information systems.

Therefore in this chapter we formalize implicit knowledge as a set of well-defined constraints, using first-order predicate logic. These constraints are an integral part of our service ontology. We implemented the majority of these constraints in our OBELIX software tools (see Section 6.1), and used them in the process of configuring service bundles. The constraints will also be implemented in DEM-DISC (see Section 6.2). Some of the constraints represent knowledge that the service configuration process takes into consideration, while others represent inconsistencies that we expect users not to cause, but a good information systems should ideally warn users who cause such inconsistencies.

Summary

We present an ontology – a formalized conceptual model – of services, with the aim to develop software for service bundling.

A service bundle consists of more elementary services, like a PC consists of smaller hardware components. Service providers can offer service bundles via the Internet. The realization of websites where customers can find and buy a service bundle that suits their needs requires that software supports service bundling. Software, in turn, can reason only about formalized knowledge. Ontologies – for example our *serviguration* service ontology – provide the necessary formality for software support.

Why is service bundling an interesting topic?
The practice of selling services as bundles has been gaining ground in recent years. *Service bundling* is the sale of two or more services as one package. From a customer perspective, the price of a bundle is often lower than the sum of the prices of the elements within the bundle. Furthermore, a service bundle can often satisfy *complex* customer needs, while the single services in the bundle fail to do so. Also from a supplier perspective, bundling presents a number of advantages, including cost efficiency, product differentiation, increasing revenues and increasing a firm's competitiveness by introducing entry barriers.

Core idea: service bundling is a component configuration task
Thus a service bundle is a complex service, including a number of more elementary services, that are packaged together such that the bundle presents added value to customers and to suppliers. A service bundle is a complex artifact, similarly to a PC that is composed of a motherboard, memory, a processor and more. Composing these components into a PC is referred to as a configuration task. A wealth of research has been performed within computer science and artificial intelligence departments about *configuration*, a task of designing a complex artifact based on a set of components, a description of how these components can be connected to each other and a description of the desired artifact.

In spite of the growing importance of the service industry in general and of service bundling in particular, so far the Internet has mainly been used to allow customers design complex *goods*, for example PCs or cars. Examples are websites of market

leaders as Dell, Cisco and BMW. In these examples a customer is guided through the design – or configuration – of a final product out of more elementary components. Such e-commerce scenarios are facilitated by a component-based description of elements that constitute the final product. Similar scenarios for services would require a formalized component-based description of services as well. Such a description does not yet exist, as research on services has so far been performed in business schools, where formal and computer-based reasoning and knowledge representation techniques are not common.

Challenge: services differ from physical goods
A number of software based product configurators exist on the market, based on the fruits of configuration research. Product configurators have two important characteristics that are not applicable for services. First, elementary components and rules for connecting these components are described in product configurators based on physical properties of the components. A main difference between services and goods is the intangibility of services. Goods are objects that one can hold in your hands and drop on the floor. Services, on the other hand, are of an intangible nature; they provide experiences and capabilities. Even when a service is accompanied by a so-called *physical evidence*, for example a transportation ticket, this physical evidence merely functions as an enhancer of customers' impression, but it is not the essence of the service itself. Second, product configurators do not take higher-level business logic (e.g., marketing considerations, competition) into consideration in the configuration process. In real-world, however, services are composed into service bundles based on business logic.

Our solution: *serviguration*
We developed a service ontology to overcome differences between service bundling and traditional configuration of physical goods. Our ontology is called *serviguration* because we use it for service configuration. We show that service bundling can be represented as a configuration task, and that software can be developed to configure service bundles, just like software configures a PC out of more elementary components. To achieve this, our ontology describes services from a business value perspective. First, instead of describing services by physical properties, we describe them by the exchange of economic values between suppliers and customers. At the same time, service description adheres also to component description, as described in literature about configuration. Second, rules (so-called 'constraints') for combining services into bundles represent a firm's or an industry's business logic, rather than constraints on how physical elements can be connected. For example, one service may substitute another, or two services may not be sold together because their suppliers are competitors who do not wish to collaborate.

We show that software configurators can in fact configure complex *intangible* artifacts – services – based on our ontology's service description. We used our ontology

and a self-developed *serviguration* software tool to configure service bundles in complex real-world studies in the energy sector and in the health sector. Currently we are involved in a project in the health sector, where a Web-based information system is being developed based on our *serviguration* ontology, that shall offer dementia patients and their informal carers customer-tailored service bundles, based on their specific needs.

A software tool we have developed for *serviguration* demonstrates another important principle in our work: different knowledge representations are required for different stakeholders. We represent our ontology using semi formal UML diagrams and a Web based knowledge representation standard (RDFS) to facilitate automation. However, domain experts who would have to actually model their services are not helped with such knowledge representation techniques. To this end, we developed a graphical representation for our service ontology. Our experience shows that a graphical notation is very suitable for communication with non-ICT stakeholders.

Samenvatting

Dit proefschrift gaat over het *automatiseren van service bundeling*. Wat houdt dat eigenlijk in?

'Bundeling' is het verkopen van meerdere producten (diensten of goederen) als één pakket. Vaak is zo'n bundel aantrekkelijk voor klanten, omdat de prijs ervan lager is dan de som van de prijzen van de losse elementen. Maar misschien nog belangrijker: een bundel kan een complexe vraag bevredigen, die niet bevredigd kan worden door losse, elementaire producten. Een goed voorbeeld hiervoor is de zorgsector, waar klanten vaak kampen met een vraag die niet door een enkele dienst beantwoord kan worden; wel door een pakket van diensten. Ook voor leveranciers brengt bundeling meerdere voordelen met zich mee. Bijvoorbeeld omdat meerdere diensten eenzelfde infrastructuur gebruiken, zodat het verlenen van twee diensten aan eenzelfde klant goedkoper is dan het verlenen van de twee zelfde diensten als losse diensten aan twee verschillende klanten. Of omdat klanten dan uiteindelijk meer kopen (en dus meer geld uitgeven) dan wanneer de losse elementen worden aangeboden. Neem het voorbeeld van *Microsoft Office Basic Suite* met Word, Excel en Outlook. Ook klanten die slechts één of twee van deze applicaties gebruiken, betalen wel voor het gehele pakket. Bundeling versterkt ook de concurrentiepositie van leveranciers. Als leveranciers een "totaal pakket" leveren aan hun klanten, wordt het moeilijker voor nieuwe concurrentie. Om concurrentie te bieden, moeten nieuwe concurrenten niet alleen één nieuwe dienst op de markt brengen, maar een geheel pakket. Zo zijn er nog meer redenen waarom bundeling een belangrijk verschijnsel wordt.

Ook het aandeel van de service industrie in het nationale product wordt aanzienlijk groter. In de VS zijn er meer banen in de service sector dan in alle andere sectoren samen. De kernactiviteit van veel economieën verandert van productie van goederen naar dienstverlening, waarbij de plaats van de "industriële revolutie" is vervangen door een "kennis revolutie"of "informatie revolutie". En soms heeft men het ook over een "Internet revolutie".

Want tegelijkertijd is het gebruik van het Internet enorm gegroeid bij bedrijven en bij particulieren. Dankzij het Internet kunnen klanten (particulieren én bedrijven) sneller, goedkoper en flexibeler (locatie- en tijdonafhankelijk) transacties uitvoeren.

Het Internet als infrastructuur maakt het ook mogelijk voor bedrijven om informatie uit te wisselen en bedrijfsprocessen te koppelen, zodat ze *samen* hun aanbod als één geheel kunnen verkopen aan klanten. Het Internet speelt dus een dubbele rol: als *facilitator* en als *enabler*. Aan de ene kant kunnen bestaande processen effectiever en efficiënter worden uitgevoerd dankzij het Internet; aan de andere kant maakt het Internet nieuwe manieren van werken mogelijk (bijvoorbeeld samenwerkingsverbanden tussen bedrijven).

Breng deze drie ontwikkelingen (de opkomst van bundeling, van de service industrie en van Internet gebruik) bij elkaar, en je komt op het online verkopen van service bundels. Oftewel *e-service bundeling*.

Hoe ontwerp je een goede bundel? Waarom zou je juist bepaalde diensten als één pakket verkopen? Dit ontwerp proces dient rekening te houden met twee perspectieven: die van de klant, en die van de leverancier. Want uiteindelijk geldt dat een bundel niet zal worden aangeboden en afgenomen, als deze bundel niet voor beide partijen interessant is. Een bundel hoort daarom ontworpen te worden zodat het een meerwaarde biedt aan de klant, en zodat het volgens de "business logica" van de leverancier(s) is samengesteld.

Een andere complicerende factor is hoe het ontwerpen van bundels geautomatiseerd kan worden, zodat het bijvoorbeeld online kan gebeuren. De kennis ("business logica") waarop het bundeling proces berust zit vaak in de hoofden van service personeel, maar het wordt nergens expliciet geformaliseerd, dus beschreven op een wiskunde- of logica gebaseerde wijze. Dit is een belangrijk obstakel bij automatiseringsprojecten, want software kan alleen redeneren over geformaliseerde kennis die gerepresenteerd wordt in een door machine interpreteerbare taal.

In dit multidisciplinaire proefschrift doen we precies dát. We presenteren een conceptueel model – een formele beschrijving – van diensten en van de logica achter het ontwerpen van service bundels, vanuit de perspectieven van klanten en van leveranciers. We passen werk- en redeneertechnieken vanuit de informatica toe op een bedrijfskundig onderwerp, met als doel om redeneerprocessen over dit bedrijfskundig onderwerp te kunnen automatiseren. Een goede oplossing vereist dat kennis en inzichten vanuit de bedrijfskunde en vanuit de informatica worden samengebracht. We beschrijven diensten zodanig dat het samenstellen van service bundels vergelijkbaar is met het bouwen van een kasteel uit lego blokjes. In tegenstelling tot traditioneel onderzoek in de bedrijfskunde, beschrijven we onze resultaten niet alleen in natuurlijke taal, maar ook in semi-formele diagrammen en in een door machine interpreteerbare Web-gebaseerde taal (RDFS). In tegenstelling tot traditioneel onderzoek in de informatica, berust ons werk inhoudelijk op kennis en inzichten vanuit de bedrijfskunde.

Kern ideeën in onze aanpak zijn:

- Als klanten een dienst afnemen, zijn ze niet geïnteresseerd in de dienst zelf, maar in de *voordelen* die deze dienst met zich meebrengt. Bijvoorbeeld, een klant is niet per se geïnteresseerd in een *vlucht* van Amsterdam naar Miami, maar in een verandering van zijn *toestand* (locatie, in dit geval). Deze voordelen hebben een economische waarde (de klant is bereid om ervoor te betalen) omdat ze *behoeften* van klanten bevredigen. Dus klanten nemen diensten af omdat deze diensten hun behoeften bevredigen.

- In tegenstelling tot goederen, die fysiek waarneembare eigenschappen hebben aan de hand waarvan ze door klanten en leveranciers ondubbelzinnig beschreven kunnen worden, hebben diensten een ontastbare aard. Een dienst kan niet worden beschreven aan de hand van zijn lengte, gewicht, of kleur, en dient daarom anders beschreven te worden.

- Diensten zijn economische activiteiten waarin klanten en leveranciers objecten met een economische waarde uitwisselen (geld, rechten, ervaringen, goederen enzovoort) zodat elk van ze meent meer waarde te hebben ontvangen dan gegeven. We beschrijven diensten daarom als een uitwisseling van economische waarden tussen klanten en leveranciers. Dit is overeenkomstig met hoe het begrip 'dienst' in de bedrijfskunde geïnterpreteerd wordt.

- Klanten en leveranciers hebben een verschillende kijk op behoeften van klanten, en gebruiken verschillende terminologiën om deze behoeften te beschrijven. Beide perspectieven dienen meegenomen te worden.

Op het moment van schrijven zijn we bezig met het FrUX project, mede-gefinancieerd door het ministerie van economische zaken, waarin ons model gebruikt wordt als basis voor een Web-gebaseerd informatiesysteem genaamd DEM-DISC dat aan dementie patiënten en aan hun mantelzorgers service bundels zal aanbieden, aan de hand van hun persoonlijke behoeften en situatie. Een groot probleem in de zorg sector is dat het zorgaanbod zeer groot en gefragmenteerd is, waardoor klanten door de bomen het bos niet zien. Dementie patiënten en hun mantelzorgers zijn vaak ouderen; voor deze doelgroep is het vaak nog moeilijker om het juiste zorgaanbod te vinden. Daarom is het belangrijk om juist deze doelgroep te ondersteunen. DEM-DISC zal kennis hebben van de verschillende diensten voor deze doelgroep, en zal gestructureerde redeneertechnieken vanuit de Kennis Management, Kunstmatige Intelligentie en Requirements Engineering gebruiken om behoeften van klanten te koppelen aan beschikbare diensten van een groot aantal leveranciers, om vervolgens deze diensten te bundelen tot een diensten pakket, een *service bundel*.

Bibliography

Aalten, P. (2004), Behavioral problems in dementia. Course and risk factors, PhD thesis, Maastricht University, Maastricht, The Netherlands.

Akkermans, H., Baida, Z., Gordijn, J., Peña, N., Altuna, A. & Laresgoiti, I. (2004), 'Value webs: Using ontologies to bundle real-world services', *IEEE Intelligent Systems - Semantic Web Services* **19**(4), 57–66.

Altuna, A., Cabrerizo, A., Laresgoiti, I., Peña, N. & Sastre, D. (2004), Co-operative and distributed configuration, *in* 'Proceedings of Net.ObjectDays (NODe) 2004', Erfurt, Germany.

Ambler, C. A. (1998), 'Developing a product classification system for the united states', *US Census Bureau* . Available from http://www.census.gov/eos/www/napcs/documentlinks2.htm, last visited December 2005.

Ambriola, V. & Gervasi, V. (1997), Processing natural language requirements, *in* 'Proceedings of the 12th International Conference on Automated Software Engineering (ASE '97)', IEEE Computer Society, Lake Tahoe, CA.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Trickovic, I. & Weerawarana, S. (2003), *Business Process Execution Language for Web Services Version 1.1*. Available from http://www-128.ibm.com/developerworks/library/specification/ws-bpel/, last visited December 2005.

Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Meyer, M., Petrone, G., Schäfer, R., Schütz, W. & Zanker, M. (2002), Personalizing on-line configuration of products and services, *in* 'Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002)', Lyon, France.

Austin, D., Barbir, A., Ferris, C. & Garg, S. (2004), 'Web services architecture requirements, w3c working group note'. http://www.w3.org/TR/wsa-reqs, last visited December 2005.

Avison, D., Lau, F., Myers, M. & Nielsen, P. A. (1999), 'Action research', *Communications of the ACM* **42**(1), 94–97.

Baida, Z. (2002), Architecture visualization, Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands.

Baida, Z., Akkermans, H. & Gordijn, J. (2003), Serviguration: Towards online configurability of real-world services, *in* 'Proceedings of the Fifth International Conference on Electronic Commerce (ICEC03)', ACM Press, Pittsburgh, PA, pp. 111–118.

Baida, Z., Akkermans, H. & Gordijn, J. (2005), Service classification versus configuration, *in* 'Proceedings of the Workshop on Product-Related Data in Information Systems (PRODIS 2005, in conjunction with Informatik 2005)', Lecture Notes in Informatics, Bonn, Germany.

Baida, Z., de Bruin, H. & Gordijn, J. (2003), 'e-business cases assessment: From business value to system feasibility', *International Journal of Web Engineering and Technology* **1**(1), 127–144.

Baida, Z., Gordijn, J., Akkermans, H., Sæle, H. & Morch, A. Z. (2005), 'Finding e-service offerings by computer-supported customer need reasoning', *International Journal of E-Business Research (IJEBR)* **1**(3), 91–112.

Baida, Z., Gordijn, J., Morch, A. Z., Sæle, H. & Akkermans, H. (2004), Ontology-based analysis of e-service bundles for networked enterprises, *in* 'Proceedings of the 17th Bled eCommerce Conference (Bled 2004)', Bled, Slovenia.

Baida, Z., Gordijn, J., Omelayenko, B. & Akkermans, H. (2004), A shared service terminology for online service provisioning, *in* 'Proceedings of the Sixth International Conference on Electronic Commerce (ICEC04)', ACM Press, Delft, The Netherlands.

Baida, Z., Gordijn, J., Sæle, H., Akkermans, H. & Morch, A. Z. (2005), An ontological approach for eliciting and understanding needs in e-services, *in* 'Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005), volume 3520 of Lecture Notes in Computer Science (LNCS)', Springer-Verlag, Porto, Portugal, pp. 400–414.

Baida, Z., Gordijn, J., Sæle, H., Morch, A. Z. & Akkermans, H. (2004), Energy services: A case study in real-world service configuration, *in* 'Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004), volume 3084 of Lecture Notes in Computer Science (LNCS)', Springer-Verlag, Riga, Latvia, pp. 36–50.

Barrutia Legarreta, J. M. & Echebarria Miguel, C. (2004), 'Collaborative relationship bundling: a new angle on services marketing', *International Journal of Service Industry Management* **15**(3), 264–283.

Baskerville, R. & Myers, M. D. (2004), 'Special issue on action research in information systems: making IS research relevant to practice – foreword', *MIS Quarterly* **28**(3), 329–335.

Benatallah, B., Sheng, Q. Z. & Dumas, M. (2003), 'The Self-Serv environment for web services composition', *IEEE Internet Computing* **7**(1), 40–48.

Bennett, R. J. & Robson, P. J. (2001), 'Exploring the market potential and bundling of business association services', *Journal of Services Marketing* **15**(3), 222–239.

Berry, L. L. & Parasuraman, A. (1991), *Marketing Services: Competing through Quality*, The Free Press, New York, NY.

Bigné, E., Martínez, C. & Miquel, M. J. (1997), The influence of motivation, experience and satisfaction on the quality of service of travel agencies, *in* P. Kunst & J. Lemmink, eds, 'Managing Service Quality (Volume III)', Paul Chapman Publishing Ltd, London, UK, pp. 53–70.

Birmingham, W. P., Brennan, A., Gupta, A. P. & Siewiorek, D. P. (1988), 'MICON: a single-board computer synthesis tool', *IEEE Circuits and Devices Magazine* **4**(1), 37–46.

Bolton, R. N. (2003), 'Marketing challenges of e-services', *Communications of the ACM* **46**(6), 43–44.

Borst, P. (1997), Construction of Engineering Ontologies for Knowledge Sharing and Reuse, PhD thesis, Universiteit Twente, Enschede, The Netherlands.

Borst, P., Akkermans, H. & Top, J. (1997), 'Engineering ontologies', *International Journal of Human-Computer Studies* **46**, 365–406.

Braekhus, A., Oksengard, A. R., Engedal, K. & Laake, K. (1998), 'Social and depressive stress suffered by spouses of patients with mild dementia', *Scandinavian Journal of Primary Health Care* **16**(4), 242–246.

Brézillon, P. (1999), 'Context in problem solving: a survey', *The Knowledge Engineering Review* **14**(1), 47–80.

Burns, A. (2000), 'The burden of alzheimer's disease', *International Journal of Neuropsychopharmacology* **3**(7), 31–38.

Carmon, Z., Shanthikumar, J. & Carmon, T. (1995), 'A psychological perspective on service segmentation models: The significance of accounting for consumers' perceptions of waiting and service', *Management Science* **41**(11), 1806–1815.

Centeno, C. (2003), 'Adoption of Internet services in the enlarged european union: Lessons from the Internet banking case'. Available at http://fiste.jrc.es/download/eur20822en.pdf, last visited December 2005.

Chan, H., Lee, R., Dillon, T. & Chang, E. (2001), *E-Commerce*, John Wiley & Sons, Chichester, UK.

Choi, S., Stahl, D. O. & Whinston, A. B. (1997), *The Economics of Electronic Commerce*, Macmillan Technical Publishing, Indianapolis, Indiana.

Cologne Institute of Business Research (2000), 'eCl@ss standardized material and service classification: Structure of the sets of attributes'. Available from www.eclass.de/informationen/download/eClassAttributes5_00.doc, last visited December 2005.

Coope, B., Ballard, C., Saad, K., Patel, A., Bentham, P., Bannister, C., Graham, C. & Wilcock, G. (1995), 'The prevalence of depression in the carers of dementia sufferers', *International Journal of Geriatric Psychiatry* **10**(3), 237–242.

Corcho, Ó. & Gómez-Pérez, A. (2001), Webpicker: Knowledge extraction from web resources, *in* 'Proceedings of the 6th International Workshop on Applications of Natural Language for Information Systems (NLDB01)', Madrid, Spain, pp. 55–64.

Cunis, R., Günter, A., Syska, I., Peters, H. & Bode, H. (1989), PLAKON – an approach to domain-independent construction, *in* 'IEA/AIE '89: Proceedings of the second international conference on Industrial and engineering applications of artificial intelligence and expert systems', ACM Press, New York, NY, USA, pp. 866–874.

Cunningham, L. F., Young, C. E., Ulaga, W. & Lee, M. (2004), 'Consumer views of service classifications in the USA and France', *Journal of Services Marketing* **18**(6), 421–432.

Daly, J. (2002), *Pricing for Profitability: Activity Based Pricing for Competitive Advantage*, John Wiley & Sons, New York, NY.

de Bruin, H. & van Vliet, H. (2002), Top-down composition of software architectures, *in* P. Runeson, ed., 'Proceedings of 9th International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS2002)', IEEE Computer Society, Lund, Sweden, pp. 1–10.

de Bruin, H., van Vliet, H. & Baida, Z. (2002), Documenting and analyzing a context-sensitive design space, *in* J. Bosch, M. Gentleman, C. Hofmeister & J. Kuusela, eds, 'Software Architecture: System Design, Development and Maintenance; Proceedings of the 3rd Working IFIP/IEEE Conference on Software Architecture (WICSA-02)', Kluwer Academic Publishers, Montreal, Canada, pp. 127–141.

de Miranda, B. (2005), Incorporating pricing models in the service ontology, Master's thesis, Vrije Universiteit, Amsterdam, The Netherlands.

de Ruyter, K., Wetzels, M. & Kleijnen, M. (2001), 'Customer adoption of e-service: an experimental study', *International Journal of Service Industry Management* **12**(2), 184–207.

de Vugt, M. E. (2004), Behavioral problems in dementia, caregiver issues, PhD thesis, Maastricht University, Department of Psychiatry and Neuropsychiatry, Maastricht, The Netherlands.

de Vugt, M. E., Stevens, F., Aalten, P., Lousberg, R., Jaspers, N., Winkens, I., Jolles, J. & Verhey, F. R. J. (2003), 'Behavioural disturbances in dementia patients and quality of the marital relationship', *International Journal of Geriatric Psychiatry* **18**(2), 149–154.

Dolan, R. J. & Simon, H. (1996), *Power pricing. How Managing Price Transforms the Bottom Line*, The Free Press, New York, NY.

Donnelly, W. A. (1999), 'International and domestic product classifications', *Office of Economics, U.S. International Trade Commission* . Document No. 99-03-A-Revised, available from www.gpo.gov, last visited January 2005.

Donzelli, P. (2004), 'A goal-driven and agent-based requirements engineering framework', *Requirements Engineering* **9**(1), 16–39.

Doulkeridis, C., Valavanis, E. & Vazirgiannis, M. (2003), Towards a context-aware service directory, *in* 'Proceedings of the 4th VLDB Workshop on Technologies for E-Services (TES'03), held in conjunction with the 29th International Conference on Very Large Data Bases (VLDB 2003)', Berlin, Germany. Available from http://www.db-net.aueb.gr/stratis/Research/Publications.htm, last visited December 2005.

Dröes, R. M., Breebaart, E., Meiland, F. J. M., van Tilburg, W. & Mellenbergh, G. (2004), 'Effect of meeting centres support program on feelings of competence of family carers and delay of institutionalization of people with dementia', *Aging & Mental Health* **8**(3), 201–211.

Dröes, R. M., Meiland, F. J. M., Doruff, C., Varodi, I., Akkermans, H., Baida, Z., Faber, E., Haaker, T., Moelaert, F., Kartseva, V. & Tan, Y.-H. (2005), A dynamic interactive social chart in dementia care. Attuning demand and supply in the care for persons with dementia and their carers, *in* L. Bos, S. Laxminarayan & A. Marsh, eds, 'Medical and Care Compunetics 2, Volume 114 Studies in Health Technology and Informatics', IOS Press, The Netherlands, USA, England, pp. 210–220.

Dumas, M., O'Sullivan, J., Heravizadeh, M., Edmond, D. & ter Hofstede, A. H. M. (2001), Towards a semantic framework for service description, *in* 'Proceedings of the IFIP Conference on Database Semantics', Kluwer Academic Publishers, Hong Kong.

Eagles, J. M., Craig, A., Rawlinson, F., Restall, D. B., Beattie, J. A. & Besson, J. A. (1987), 'The psychological well-being of supporters of the demented elderly', *British Journal of Psychiatry* **150**, 293–298.

eCl@ss (2000), 'ecl@ss white paper, version 0.6'. Available from www.eclass.de/informationen/download/eClassWhitePaper06.doc, last visited December 2005.

eCl@ss website (2005). http://www.eclass.de.

ECPC Issue Paper 1 (1993), *US Federal Register* **58**(60), 16991–17000.

Edmond, D. & ter Hofstede, A. H. M. (2000), Service composition for electronic commerce, *in* 'Proceedings of PACIS-2000 (Pacific Asia Conference on Information Systems)', Hong Kong.

Edvardsson, B., Gustafsson, A. & Roos, I. (2005), 'Service portraits in service research: a critical review', *International Journal of Service Industry Management* **16**(1), 107–121.

Essegaier, S., Gupta, S. & Zhang, Z. J. (2002), 'Pricing access services', *marketing Science* **21**(2), 139–159.

Flæte, A. & Ottesen, G. (2001), *Telefiasko for Viken*, Dagens Næringsliv (newspaper), 13/14 October 2001, Norway.

Fowler, M. & Scott, K. (1997), *UML Distilled: Applying The Standard Object Modeling Language*, Addison-Wesley, Chichester, UK.

Fox, G., Lantner, K. & Marcom, S. (1997), A software development process for COTS-based information system infrastructure, *in* 'Proceedings of 5th International Symposium on Assessment of Software Tools (SAST '97)', IEEE.

Fox, M. S. & Gruninger, M. (1998), 'Enterprise modelling', *AI Magazine* **19**(3), 109–121.

Fuxman, A., Liu, L., Pistore, M., Roveri, M. & Mylopoulos, J. (2003), Specifying and analyzing early requirements: Some experimental results, *in* 'Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03)', IEEE Computer Society, Monterey Bay, California, pp. 105–114.

Geerts, G. & McCarthy, W. (1999), 'An accounting object infrastructure for knowledge-based enterprise models', *IEEE Intelligent Systems and Their Applications* **14**(4), 89–94.

Glushko, R. J., Tenenbaum, J. M. & Meltzer, B. (1999), 'An XML Framework for Agent-Based E-commerce', *Communications of the ACM* **42**(3), 106–114.

Gómez-Pérez, A., González-Cabero, R. & Lama, M. (2004), 'ODE SWS: A framework for designing and composing semantic web services', *IEEE Intelligent Systems - Semantic Web Services* **19**(4), 24–31.

Gordijn, J. (2002), Value-based Requirements Engineering: Exploring Innovative e-Commerce Ideas, PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands.

Gordijn, J. & Akkermans, H. (2001), 'Designing and evaluating e-Business models', *IEEE Intelligent Systems - Intelligent e-Business* **16**(4), 11–17.

Gordijn, J. & Akkermans, H. (2003*a*), Does e-business modeling really help?, *in* 'Proceedings of the 36th Hawaii International Conference On System Sciences', IEEE.

Gordijn, J. & Akkermans, H. (2003*b*), 'Value based requirements engineering: Exploring innovative e-commerce ideas', *Requirements Engineering Journal* **8**(2), 114–134.

Gordijn, J., Akkermans, H. & van Vliet, H. (2000), Business modelling is not process modelling, *in* 'Conceptual Modeling for E-Business and the Web (eCOMO2000), volume 1921 of Lecture Notes in Computer Science (LNCS)', Springer-Verlag, Salt Lake City, USA, pp. 40–51.

Govindarajan, K., Karp, A., Kuno, H., Beringer, D. & Banerji, A. (2001), 'Conversation definitions: defining interfaces of web services', *In HP Position Papers for the World Wide Web Consortium (W3C) Workshop on Web Services, Hewlett-Packard report HPL-2001-73* . Available from http://www.hpl.hp.com/techreports/2001/HPL-2001-73.html, last visited December 2005.

Granada Research (1998), 'Using the UNSPSC: Why coding and classifying products is critical to success in electronic commerce', *White paper* . The document was first written in 1998, and updated in 2001. Available from www.unspsc.org/AdminFolder/Documents/UNSPSC_White_Paper.doc, last visited December 2005.

Greene, J. G., Smith, R., Gardiner, M. & Timbury, G. C. (1980), 'Measuring behavioural disturbance of elderly demented patients in the community and its effects on relatives: a factor analytic study', *Age Ageing* **11**(2), 121–126.

Grönroos, C. (2000), *Service Management and Marketing: A Customer Relationship Management Approach, 2nd edition*, John Wiley & Sons, Chichester, UK.

Gruber, T. (2004), 'Interview with Tom Gruber', *Bulletin of AIS Special Interest Group on Semantic Web and Information Systems* **1**(3).

Gruber, T., Olsen, G. & Runkel, J. (1996), 'The configuration design ontologies and the VT elevator domain theory', *International Journal of Human-Computer Studies* **44**(3/4), 569–598.

Gruber, T. R. (1995), 'Towards principles for the design of ontologies used for knowledge sharing', *International Journal of Human-Computer Studies* **43**, 907–928.

Guiltinan, J. P. (1987), 'The price bundling of services: a normative framework', *Journal of Marketing* **51**(2), 74–85.

Günter, A. (1992), Flexible Control in Expert Systems for Planning and Configuration in Technical Domains, PhD thesis.

Gutman, J. (1982), 'A means-end chain model based on consumer categorization processes', *Journal of Marketing* **46**(2), 60–72.

Health Council of the Netherlands (2002), *Dementia*, number 2002/04, The Hague, The Netherlands.

Heinrich, M. (1991), Ressourcen-orientierte modellierung als basis des konfigurierens modularer technischer systeme, *in* 'Proceedings of the 5th Workshop Planen und Konfigurieren', Hamburg, Germany, pp. 61–74.

Heinrich, M. & Jüngst, E. W. (1991), A resource-based paradigm for the configuring of technical systems from modular components, *in* 'Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications (CAIA)', pp. 257–264.

Heylighen, F. (2001), 'Bootstrapping knowledge representations: From entailment meshes via semantic nets to learning webs', *Kybernetes* **30**(5/6), 691–722.

Hill, T. P. (1977), 'On goods and services', *Review of Income and Wealth* **23**(4), 315–338.

Holbrook, M. B. (1999), *Consumer Value: A Framework for Analysis and Research*, Routledge, New York, NY.

Hotle, M. (2003), 'A conceptual evolution: From process to web services', *Gartner Group Research Note TU-16-1420* .

Hull, R., Benedikt, M., Christophides, V. & Su, J. (2003), E-services: a look behind the curtain, *in* 'Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems', ACM Press, pp. 1–14.

Hunt, S. (1976), *Marketing Theory*, Grid, Columbus, OH.

IBM (2000), 'Web services: Taking e-business to the next level, white paper'. Available from http://www-900.ibm.com/developerWorks/cn/wsdd/download/pdf/e-businessj.pdf, last visited December 2005.

IDEF0 website (2005). http://www.idef.com/pdf/idef0.pdf.

Jain, P. & Schmidt, D. C. (1997), Service configurator: A pattern for dynamic configuration of services, *in* 'Proceedings of the Third USENIX Conference on Object-Oriented Technologies (COOTS 1997)', Portland, Oregon.

Janda, S., Trocchia, P. J. & Gwinner, K. P. (2002), 'Consumer perceptions of internet retail service quality', *International Journal of Service Industry Management* **13**(5), 412–431.

Johannesson, P., Wangler, B. & Jayaweera, P. (2000), Application and process integration - concepts, issues, and research directions, *in* S. Brinkkemper, E. Lindencrona & A. Slvberg, eds, 'Information Systems Engineering Symposium CAiSE 2000', Springer Verlag, Chicago.

Kalfoglou, Y. & Schorlemmer, M. (2003), 'Ontology mapping: the state of the art', *The Knowledge Engineering Review* **18**(1), 1–31.

Kasper, H., van Helsdingen, P. & de Vries jr, W. (1999), *Service Marketing Management: An International Perspective*, John Wiley & Sons, Chichester, UK.

Kaufer, D. I., Cummings, J. L., Christine, D., Bray, T., Castellon, S., Masterman, D., MacMillan, A., Ketchel, P. & DeKosky, S. T. (1998), 'Assessing the impact of neuropsychiatric symptoms in alzheimer's disease: the neuropsychiatric inventory caregiver distress scale', *Journal of American Geriatric Society* **46**(2), 210–215.

Keck, D. O. & Kuehn, P. J. (1998), 'The feature and service interaction problem in telecommunications systems: A survey', *IEEE Transactions on Software Engineering* **24**(10), 779–796.

Keller, G., Nüttgens, M. & Scheer, A. W. (1992), Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)", Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany.

Klein, M. & Dallarocas, C. (1999), Exception handling in agent systems, *in* O. Etzioni, J. P. Müller & J. M. Bradshaw, eds, 'Proceedings of the Third International Conference on Autonomous Agents (Agents'99)', ACM Press, Seattle, WA, USA, pp. 62–68.

Kopisch, M. & Günter, A. (1992), Configuration of a passenger aircraft cabin based on conceptual hierarchy, constraints and flexible control, *in* 'Proceedings of the 5th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems', Springer Verlag, pp. 421–430.

Kotler, P. (1980), *Principles of Marketing*, Prentice Hall, Englewood Cliffs, NJ.

Kotler, P. (1988), *Marketing Management: Analysis, Planning, Implementation and Control, 6th edition*, Prentice Hall, Englewood Cliffs, NJ.

Kotov, V. (2001), 'Towards service-centric system organization, Hewlett-Packard report HPL-2001-54'. Available from http://www.hpl.hp.com/techreports/2001/HPL-2001-54.html, last visited December 2005.

Koutsopoulou, M., Farmakis, C. & Gazis, E. (2001), Subscription management and charging for value added services in UMTS networks, *in* 'IEEE Semiannual Vehicular Technology Conference VTC2001'.

Kreger, H. (2003), 'Fulfilling the web service promise', *Communications of the ACM* **46**(6), 29–34.

Lalioti, V. & Loucopoulos, P. (1994), 'Visualization of conceptual specifications', *Information Systems* **19**(3), 291–309.

Lancaster, K. J. (1966), 'A new approach to consumer theory', *Journal of Political Economy* **74**(2), 132–157.

Lenk, K. (1995), 'Contracts, multimedia networks and the human interface: Enhancing public and commercial services through small common serviceshops', *Computers, Environment and Urban Systems* **19**(3), 193–200.

Levitt, T. (1973), *Your Factories in the Field: Customer Service and Service Industries*, McGraw-Hill, New York, chapter 3, in Marketing for Business Growth, pp. 51–70.

Liljander, V. & Strandvik, T. (1997), 'Emotions in service satisfaction', *International Journal of Service Industry Management* **8**(2), 148–169.

Löckenhoff, C. & Messer, T. (1994), Configuration, *in* J. Breuker & W. Van de Velde, eds, 'CommonKADS Library for Expertise Modelling – Reusable Problem Solving Components, Chapter 9', IOS Press, Amsterdam, The Netherlands, pp. 197–212.

Lovelock, C. (1983), 'Classifying services to gain strategic marketing insights', *Journal of Marketing* **47**(3), 9–20.

Lovelock, C. (2001), *Services Marketing, People, Technology, Strategy, 4th edition*, Prentice Hall, Englewood Cliffs, NJ.

Malone, T. W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Osborn, J. Q. C. S., Bernstein, A., Herman, G., Klein, M. & ODonnell, E. (1999), 'Tools for inventing organizations: Toward a handbook of organizational processes', *Management Science* **45**(3), 425–443.

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. & Sycara, K. (2005), Bringing semantics to web services: The OWL-S approach, *in* 'Proceedings of the First International Semantic Web Services and Web Process Composition Workshop, volume 3387 of Lecture Notes in Computer Science (LNCS)', Springer-Verlag, San Diego, CA, USA, pp. 26–42.

Martin, J. S. & Marion, R. (2005), 'Higher education leadership roles in knowledge processing', *The Learning Organization* **12**(2), 140–151.

Martinussen, K. F. (2002), 'Kankan som kunne... (presentation at the ITEnergi 2002 conference, Bergen, Norway)'. Available from http://www.itenergi.com/kari_martinussen.ppt, last visited December 2005.

Matzler, K. (2002), 'The factor structure of service satisfaction', *International Journal of Service Industry Management* **13**(4), 314–332.

McCarthy, W. E. (1982), 'The REA accounting model: A generalized framework for accounting systems in a shared data environment', *The Accounting Review* **57**(3), 554–578.

McDermott, J. (1982), 'R1: A rule-based configurer of computer systems', *Artificial Intelligence* **19**(1), 39–88.

McIlraith, S., Son, T. & Zeng, H. (2001), 'Semantic web services', *IEEE Intelligent Systems (Special Issue on the Semantic Web)* **16**(2), 46–53.

Meerveld, J., Schumacher, J., Krijger, E., Bal, R. & Nies, H. (2004), 'Landelijk dementieprogramma, werkboek'. Available from http://www.dementieprogramma.nl, last visited December 2005.

Meiland, F. J., Danse, J. A., Wendte, J. F., Klazinga, N. S. & Gunning-Schepers, L. J. (2001), 'Caring for relatives with dementia–caregiver experiences of relatives of patients on the waiting list for admission to a psychogeriatric nursing home in The Netherlands', *Scandinavian Journal of Public Health* **29**(2), 113–121.

Mendling, J., Neumann, G., & Nüttgens, M. (2005), Yet another event-driven process chain, *in* W. M. P. van der Aalst, B. Benatallah, F. Casati & F. Curbera, eds, 'Proceedings of the 3rd International Conference on Business Process Management (BPM 2005), volume 3649 of Lecture Notes in Computer Science (LNCS)', Springer Verlag, Nancy, France, pp. 428–433.

Mirakhur, A., Craig, D., Hart, D. J., McLlroy, S. P. & Passmore, A. P. (2004), 'Behavioural and psychological syndromes in alzheimer's disease', *International Journal of Geriatric Psychiatry* **19**(11), 1035–1039.

Mittal, S. & Frayman, F. (1989), Towards a generic model of configuration tasks, *in* 'Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)', Morgan Kaufmann, San Francisco, CA, pp. 1395–1401.

Mohan, K. & Ramesh, B. (2003), Ontology-based support for variability management in product and service families, *in* 'Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS 2003)', Hawaii.

Mohr, M. F. (1999), 'Identification and classification of service products: Phase I of initiative to create a North American product classification system', *US Bureau of the Census; Paper prepared for Census Advisory Committee of Professional Associations Meeting April 22-23, 1999* . Available from http://www.census.gov/eos/www/napcs/papers/cacapr99.pdf, last visited December 2005.

Mohr, M. F. (2003*a*), 'Discussion paper on developing a classification system for products produced by service industries: Issues and insights', *US Census Bureau* . The document was first written in 1998, and revised in 2003. Available from http://www.census.gov/eos/www/napcs/papers/dispap1298rev503.pdf, last visited December 2005.

Mohr, M. F. (2003*b*), 'NAPCS structure illustration: Possible product groups, sub-groups, and classes', *US Bureau of the Census* . Available from

http://www.census.gov/eos/www/napcs/papers/structured.pdf, last visited December 2005.

Mohr, M. F. & Russell, A. S. (2001), 'North American product classification system: Overview, status, and process, the U.S. perspective', *Paper prepared for the 16th Annual Meeting of the Voorburg Group on Service Statistics* . Available from http://www.voorburg.scb.se/Voorburg%20Group_NAPCS%20Process.pdf, last visited December 2005.

Morch, A. Z., Sæle, H., Baida, Z. & Foss, S. I. (2005), New approach for retail of electricity and additional services on a deregulated power market, *in* 'Proceedings of the 8th IASTED International Conference on Power and Energy Systems (PES 2005)', ACTA Press, Marina del Rey, CA.

Motta, E. (1999), *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*, IOS Press, Amsterdam, The Netherlands.

Mourdoukoutas, P. & Mourdoukoutas, P. (2004), 'Bundling in a semi-global economy', *European Business Review* **16**(5), 522–530.

Mylopoulos, J. (1992), Conceptual modeling and Telos, *in* 'Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development', Wiley, New York, NY, pp. 49–68.

Najmann, O. & Stein, B. (1992), A theoretical framework for configuration, *in* F. Belli & F. J. Radermacher, eds, 'Proceedings of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: 5th International Conference, IEA/AIE-92, volume 604 of Lecture Notes in Computer Science (LNCS)', Springer Verlag, pp. 441–450.

Normann, R. (2001), *Service Management: Strategy and Leadership in Service Business*, 3rd edn, John Wiley & Sons, Chichester, UK.

Normann, R. & Ramirez, R. (1994), *Designing Interactive Strategy: From Value Chain to Value Constellation*, John Wiley & Sons, Chichester, UK.

Omelayenko, B. (2005), Web-Service Configuration on the Semantic Web: Exploring how Semantics meets Pragmatics, PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands.

Osterwalder, A. (2004), The Business Model Ontology: A Proposition in a Design Science Approach, PhD thesis, University of Lausanne, Lausanne, Switzerland.

Osterwalder, A. & Pigneur, Y. (2002), An e-business model ontology for modeling e-business, *in* 'Proceeding of the 15th Bled Electronic Commerce Conference', Bled, Slovenia.

O'Sullivan, J., Edmond, D. & ter Hofstede, A. H. M. (2002*a*), 'Service description: A survey of the general nature of services, report FIT-TR-2003-02'. Available from http://www.bpm.fit.qut.edu.au/about/docs/service_description.pdf, last visited December 2005.

O'Sullivan, J., Edmond, D. & ter Hofstede, A. H. M. (2002*b*), 'What's in a service? towards accurate description of non-functional service properties', *Distributed and Parallel Databases* **12**, 117–133.

Oude Luttighuis, P., Lankhorst, M., van de Wetering, R., Bal, R. & van den Berg, H. (2001), 'Visualising business processes', *Computer Languages* **27**, 39–59.

OWL Services Coalition (2004), 'OWL-S: Semantic markup for web services'. http://www.daml.org/services/owl-s/1.1/.

Paolucci, M., Kawamura, T., Payne, T. & Sycara, K. (2002), Semantic matching of web services capabilities, *in* 'Proceedings of the 1st International Semantic Web Conference (ISWC 2002)', Springer-Verlag, Sardinia, Italy, pp. 333–347.

Paolucci, M., Sycara, K. & Kawamura, T. (2002), Delivering semantic web services, *in* 'Proceedings of the 12th World Wide Web Conference (WWW2003)', Budapest, Hungary, pp. 111–118.

Parasuraman, A., Zeithaml, V. A. & Malhotra, A. (2005), 'E-S-QUAL a multiple-item scale for assessing electronic service quality', *Journal of Service Research* **7**(3), 213–233.

Parsons, J. & Cole, L. (2005), 'What do the pictures mean? guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques', *Data & Knowledge Engineering* **55**(3), 327–342.

Payne, A. (1993), *The Essence of Services Marketing*, Prentice Hall, Englewood Cliffs, NJ.

Payne, T. R., Paolucci, M. & Sycara, K. (2001), Advertising and matching daml-s service descriptions, *in* 'Semantic Web Working Symposium (SWWS)', Stanford University, California.

Pedrinaci, C., Baida, Z., Akkermans, H., Bernaras, A., Gordijn, J. & Smithers, T. (2005), Music rights clearance: Business analysis and delivery, *in* 'Proceedings of the 6th International Conference on Electronic Commerce and Web Technologies (EC-Web 2005), volume 3590 of Lecture Notes in Computer Science (LNCS)', Springer-Verlag, Copenhagen, Denmark, pp. 198–207.

Pires, P. F., Benevides, M. & Mattoso, M. (2002), Building reliable web services compositions, *in* 'Net.Object Days - WS-RSD'02', pp. 551–562.

Pitta, D. A. & Laric, M. V. (2004), 'Value chains in health care', *Journal of Consumer Marketing* **121**(7), 451–464.

Pohjola, M. (2002), 'The new economy: facts, impacts and policies', *Information Economics and Policy* **14**, 133–144.

Porter, M. E. (1980), *Competitive Strategy: Techniques for Analyzing Industries and Competitors*, The Free Press, New York, NY.

Porter, M. E. (1985), *Competitive Advantage: Creating and Sustaining Superior Performance*, The Free Press, New York, NY.

Pot, A. M. (1996), Caregivers' perspectives; a longitudinal study on the psychological distress of informal caregivers of demented elderly, PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands.

Presley, A., Sarkis, J., Barnett, W. & Liles, D. (2001), 'Engineering the virtual enterprise: An architecture-driven modeling approach', *International Journal of Flexible Manufacturing Systems* **13**, 145–162.

Rathmell, J. M. (1966), 'What is meant by services?', *Journal of Marketing* **30**(4), 32–36.

Roche, C. (2003), Ontology : a survey, *in* '8th Symposium on Automated Systems Based on Human Skill and Knowledge (IFAC2003)', Göteborg , Sweden.

RosettaNet consortium (2003), 'RosettaNet and web services: An executive-level view of RosettaNet and an emerging model for application-based B2B commerce', *Technical white paper* . Available via www.rosettanet.org, last visited December 2005.

RosettaNet website (2005). http://www.rosettanet.org.

Rust, R. T. & Kannan, P. (2003), 'E-service: a new paradigm for business in the electronic environment', *Communications of the ACM* **46**(6), 36–42.

Sabin, D. & Weigel, R. (1998), 'Product configuration frameworks – a survey', *IEEE Intelligent Systems* **13**(4), 42–49.

Salzmann, C. & Schätz, B. (2003), Service based software specification, *in* 'Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACOS) ETAPS 2003', Warsaw, Poland.

Sasser, W. E., Olsen, R. P. & Wyckoff, D. D. (1978), *Management of Service Operations: Text, Cases, and Readings*, Allyn & Bacon.

Schreiber, A. T., Akkermans, J. M., Anjewierden, A. A., de Hoog, R., Shadbolt, N., van der Velde, W. & Wielinga, B. J. (2000), *Knowledge Engineering and Management: The CommonKADS Methodology*, The MIT Press, Cambridge, MA.

Schwanke, A. & Benert, J. P. (1990), Resourcenorientierte kongurierung von kommunikationssystemen, *in* 'Proceedings of the 4th Workshop Planen und Konfigurieren', Ulm, Germany.

Schweiger, J. (1992), Generating configuration expert systems from conceptual specifications of the expert knowledge, *in* 'Proceedings of the 6th European Knowledge Acquisition Workshop on Current Developments in Knowledge Acquisition (EKAW '92)', Springer-Verlag, pp. 191–210.

Schwenzfeger, C., Dörner, H., Schädler, H. & Schädler, K. (1992), Chemisches konfigurieren, *in* 'Proceedings of the Workshop Planen und Konfigurieren', Munich, Germany.

Scozzi, B. & Garavelli, C. (2005), 'Methods for modeling and supporting innovation processes in SMEs', *European Journal of Innovation Management* **8**(1), 120–137.

Shostack, G. L. (1977), 'Breaking free from product marketing', *Journal of Marketing* **41**(2), 73–80.

Sirin, E., Parsia, B. & Hendler, J. (2004), 'Filtering and selecting semantic web services with interactive composition techniques', *IEEE Intelligent Systems* **19**(4), 42–49.

Soininen, T., Tiihonen, J., Männistö, T. & Sulonen, R. (1998), 'Towards a general ontology of configuration', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **12**, 357–372.

Stafford, T. F. (2003), 'E-services: Introduction', *Communications of the ACM* **46**(6), 26–28.

Statnett SF (2005), *Production and consumption of electricity in Norway*. Available from the website of Statnett SF (Norway's national Transmission System Operator), http://www.statnett.no/default.aspx?ChannelID=1366, last visited December 2005.

Stefik, M. (1995), *Introduction to Knowledge Systems*, Morgan Kaufmann, San Francisco, CA.

Stencil Group (2001), *Defining Web Services*, The Stencil Group. www.stencilgroup.com/ideas_scope_200106wsdefined.html, last visited March 2004.

Stephens, S. A. & Tripp, L. L. (1978), Requirements expression and verification aid, *in* 'Proceedings of the 3rd international conference on Software engineering (ICSE 1978)', Atlanta, Georgia.

Stremersch, S. & Tellis, G. J. (2002), 'Strategic bundling of products and prices: A new synthesis for marketing', *Journal of Marketing* **66**(1), 55–72.

Studer, R., Benjamins, V. & Fensel, D. (1998), 'Knowledge engineering, principles, and methods', *Data and Knowledge Engineering* **25**(1–2), 161–197.

Styhre, A. & Sundgren, M. (2005), 'Action research as experimentation', *Systemic Practice and Action Research* **18**(1), 53–65.

Tank, W. (1992), Modellierung von Expertise über Konfigurierungsaufgaben, PhD thesis.

Taylor, S. A. & Hunter, G. L. (2002), 'The impact of loyalty with e-CRM software and e-services', *International Journal of Service Industry Management* **13**(5), 452–474.

Teare, R. E. (1998), 'Interpreting and responding to customer needs', *Journal of Workplace Learning* **10**(2), 76–94.

ten Teije, A., van Harmelen, F., Schreiber, A. T. & Wielinga, B. J. (1998), 'Construction of problem-solving methods as parametric design', *International Journal of Human-Computer Studies* **49**(4), 363–389.

Teri, L. (1997), 'Behavior and caregiver burden: behavioral problems in patients with alzheimer disease and its association with caregiver distress', *Alzheimer Disease and Associated Disorders* **11, Supplement 4**, S35–S38.

Tidwell, D. (2000), 'Web services – the web's next revolution, IBM tutorial'. Available from https://www6.software.ibm.com/developerworks/education/wsbasics/wsbasics-a4.pdf, last visited December 2005.

Timmermans, J. M. (2003), *Mantelzorg; over de hulp van en aan mantelzorgers*, Sociaal en Cultureel Planbureau, The Hague, The Netherlands.

Top, J. & Akkermans, H. (1994), Engineering modelling, *in* J. Breuker & W. Van de Velde, eds, 'CommonKADS Library for Expertise Modelling – Reusable Problem Solving Components, Chapter 12', IOS Press, Amsterdam, The Netherlands, pp. 265–304.

TrønderEnergi AS (2004), 'Annual report'. Available from http://www.tronderenergi.no, last visited December 2005.

Tut, M. & Edmond, D. (2002), The use of patterns in service composition, *in* 'Proceedings of the Workshop on Web Services, e-Business, and the Semantic Web, held in conjunction with CAiSE02', Springer Verlag, Toronto, Canada. Lecture Notes in Computer Science 2512.

United Nations (2002), *Central Product Classification, CPC Version 1.1*, statistical papers, series M, no. 77, ver. 1.1 edn, New York. Available from http://unstats.un.org/unsd/statcom/doc02/cpc.pdf, last visited December 2005.

UNSPSC website (2005). http://www.unspsc.org.

Uschold, M., King, M., Moralee, S. & Zorgios, Y. (1998), 'The enterprise ontology', *The Knowledge Engineering Review* **13**(1), 31–89.

van Eijk, R., Mulder, I., ter Hofte, H. & Steen, M. (2004), 'We-centric, context-aware, adaptive mobile service bundles'. Available from https://doc.telin.nl/dscgi/ds.py/ViewProps/File-46575, last visited December 2005.

van Halteren, A., Nieuwenhuis, L., Schenk, M. & Wegdam, M. (1999), Value added web: Integrating WWW with a TINA service management platform, *in* 'Proceedings of TINA Conference, IEEE', Hawaii, USA, pp. 14–23.

van Hee, K. (1994), *Information Systems Engineering – A formal approach.*, Cambridge University Press, Cambridge.

van Lamsweerde, A. (2000), Requirements engineering in the year 00: a research perspective, *in* 'Proceedings of the 22nd international conference on Software engineering', ACM Press, Limerick, Ireland, pp. 5–19.

van Lamsweerde, A. (2001), Goal-oriented requirements engineering: A guided tour, invited minitutorial, *in* 'Proceedings of RE'01 - International Joint Conference on Requirements Engineering', IEEE, Toronto, Canada, pp. 249–263.

van Lamsweerde, A., Darimont, R. & Letier, E. (1998), 'Managing conflicts in goal-driven requirements engineering', *IEEE Transactions on Software Engineering* **24**(11), 908–926.

van Riel, A. C. R., Liljander, V. & Jurriëns, P. (2001), 'Exploring consumer evaluations of e-services: a portal site', *International Journal of Service Industry Management* **12**(4), 359–377.

van Splunter, S., Sabou, M., Brazier, F. & Richards, D. (2003), Configuring web services, using structuring and techniques from agent configuration, *in* 'Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI03)', IEEE, Halifax, Canada.

Vasarhelyi, M. & Greenstein, M. (2003), 'Underlying principles of the electronization of business: a research agenda', *International Journal of Accounting Information Systems* **4**(1), 1–25.

Wielinga, B. J. & Schreiber, A. T. (1997), 'Configuration-design problem solving', *IEEE Intelligent Systems* **12**(2), 49–56.

Wieringa, R. (1998), 'A survey of structured and object-oriented software specification methods and techniques', *ACM Computing Surveys* **30**(4), 459–527.

Wieringa, R. & Dubois, E. (1998), 'Integrating semi-formal and formal software specification techniques', *Information Systems* **23**(3/4), 159–178.

World Trade Organization (2003), 'An assessment of services trade and liberalization in the united states and developing economies', *World Trade Organization, Council for Trade in Services* . Document TN/S/W/12, available from http://docsonline.wto.org, last visited December 2005.

Xue, M., Harker, P. T. & Heim, G. R. (2003), 'Incorporating the dual customer roles in e-service design'. Available at http://fic.wharton.upenn.edu/fic/papers/03/0304.pdf, last visited December 2005.

Yang, J. (2003), 'Web service componentization', *Communications of the ACM* **46**(10), 35–40.

Zarit, S. M., Reever, K. E. & Bach-Peterson, J. (1980), 'Relatives of the impaired elderly: Correlates of feelings of burden', *The Gerontologist* **20**(6), 649–655.

Zeithaml, V. A. (1988), 'Consumer perceptions of price, quality, and value: A means–end model and synthesis of evidence', *Journal of Marketing* **52**(3), 2–22.

Zeithaml, V. A. & Bitner, M. J. (1996), *Services Marketing*, McGraw-Hill, New York, NY.

Zeithaml, V. A., Parasuraman, A. & Berry, L. (1990), *Delivering Quality Service: Balancing Customer Perceptions and Expectations*, The Free Press, New York, NY.

Zeithaml, V., Parasuraman, A. & Malhotra, A. (2000), *A Conceptual Framework for Understanding E-Service Quality: Implications for Future Research and Managerial Practice, Report No. 00-115*, Marketing Science Institute, Cambridge, MA.

Zhu, K. & MacQuarrie, B. (2003), 'The economics of digital bundling: The impact of digitization and bundling on the music industry', *Communications of the ACM* **46**(9), 264–270.

Subject index

Author index

SIKS Dissertation Series

1998

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W. Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification Using Decision Trees and Neural Nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (KPN Research)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek Wijngaards (VU)
Re-design of Compositional Systems

1999-7 David Spelt (UT)
Verification Support for Object Database Design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation

2000

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UvA)
Sociaal-Organisatorische Gevolgen van Kennistechnologie; een Procesbenadering en Actorperspectief

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-Theoretic Planning of Clinical Patient Management

2000-8 Veerle Coupé (EUR)
Sensitivity Analysis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

2001

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as Problem Solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-Based User Interface Design

2001-7 Bastiaan Schönhage (VU)
DIVA: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics

2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a Multi-Agent Modeling and Simulation Language for Work Practice Analysis and Design

2001-11 Tom van Engers (VU)
Knowledge Management: The Role of Mental Models in Business Systems Design

2002

2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)
The Private Cyberspace: Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel (KUB)
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (UvA)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent System

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UvA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environment

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UvA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems – Incremental Maintenance of Indexes to Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timi

2003-18 Levente Kocsis (UM)
Learning Search Decisions

2004

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UvA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek (UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politile gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UvA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU))
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

2005

2005-01 Floor Verdenius (UvA)
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM)
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UvA)
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UvA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)
Situated Representation

2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives

2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006

2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting

2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UvA)
The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)
Building Web Service Ontologies

2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines